Coral Calero
Francisco Ruiz
Mario Piattini (Eds.)

Ontologies for Software Engineering and Software Technology

Springer

Coral Calero · Francisco Ruiz · Mario Piattini (Eds.)

Ontologies for Software Engineering and Software Technology

With 84 Figures and 46 Tables

Springer

Editors

Coral Calero
Francisco Ruiz
Mario Piattini

E.S. Informática.
Paseo de la Universidad 4
13071 Ciudad Real, Spain

Coral.Calero@uclm.es
Francisco.RuizG@uclm.es
Mario.Piattini@uclm.es

Library of Congress Control Number: 2006932286

ACM Computing Classification (1998): D.2, H.1, I.2

ISBN-10 3-540-34517-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-34517-6 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springer.com

Typesetting: by the Editors
Production: LE-TEX Jelonek, Schmidt & Vöckler GbR, Leipzig
Cover: KünkelLopka, Heidelberg

Printed on acid-free paper 45/3100YL - 5 4 3 2 1 0

Preface

Overview

Two important challenges facing current communities of researchers and practitioners in the field of software engineering and technology (SET) are knowledge integration and computer-based automatic support. The first challenge implies wasting a lot of time and effort and this is due to one of the difficulties in human relationships, namely the lack of explicit knowledge shared among members of a group/project, with other groups and with other stakeholders. The second challenge arises because many projects include the design/construction of advanced tools for supporting different software engineering activities. These tools should provide as much functionality as possible with the smallest cost of development.

Both challenges can be better and more easily approached by using ontologies. In this book, we will mainly deal with two of the multiple applications of ontologies in software engineering and technology that have been identified in the literature: (1) sharing knowledge of the problem domain and using a common terminology among all the interested people (not just researchers); and (2) filtering the knowledge when defining models and metamodels.

The utility of the first application is obvious. However, it is important and convenient to pay it opportune attention. Communication is one of the main activities (regarding duration and impact) in software projects. It is proven that participants in projects have a different knowledge of the problem domain and/or use different languages. The ambiguity of the natural language implies mistakes and nonproductive efforts. Ontologies can mitigate these problems and, farther, some authors have intended to use ontologies as back-bone of software tools and environments.

The second application is focused on the filtering of knowledge of a given domain. Models and metamodels are abstract representations of reality and, by definition, they only include a part of the reality they are aimed at modeling, obviating the unwanted characteristics. In this sense, ontologies can also help us decide what must be extracted from the real systems

to build models or what must be taken into account when defining metamodels.

So, this book should not be considered as a book written by ontology experts for ontology experts, but one written by people who use the ontologies mainly for the two applications mentioned above. For that reason, this book is oriented to researchers and practitioners in SET and includes the advanced trends in the use of ontologies within software projects and software engineering research. It also deals with two main challenges the SET discipline: (1) knowledge integration and (2) design of more powerful and generic tools.

Organization

The book is composed of eleven chapters structured into three parts: an introductory part; a part composed of ontologies that conceptualize a SET domain or subdomain; and a part where some proposals on the use of ontologies as software artifacts in some software processes and technologies are described.

The last introductory part comprises two chapters. The first one, written by Oscar Corcho, Mariano Fernández-López and Asunción Gómez-Pérez, will introduce the ontologies' concepts and the main aspects related to ontological engineering. The second chapter (by Francisco Ruiz and José R. Hilera) will deal with the state of the art of the use of ontologies in SET. Also, this chapter defines a taxonomy for classifying the uses of ontologies in SET, together with the result of the classification into this taxonomy of about 50 ontologies (including the proposals of this book).

The second part is made up of five chapters. Chapter 3 will present the engineering of the ontology for the Software Engineering Body of Knowledge, written by Alain Abran, Juan-José Cuadrado, Elena García-Barriocanal, Olavo Mendes, Salvador Sánchez-Alonso and Miguel-Angel Sicilia. An ontology for software development methodologies and endeavours will be presented by Cesar Gonzalez-Perez and Brian Henderson-Sellers in Chap. 4. Chapter 5 presents a software maintenance ontology developed by Nicolas Anquetil, Káthia M. de Oliveira and Márcio G.B. Dias, and an ontology for software measurement by Manuel F. Bertoa, Antonio Vallecillo and Félix García is the topic of Chap. 6. An ontological approach to the SQL:2003 developed by Coral Calero and Mario Piattini will be explained in Chap. 7, closing this second part.

The final part begins with the Object Management Group Ontology Definition Metamodel (Chap. 8), developed by Robert Colomb, Kerry Raymond, Lewis Hart, Patrick Emery, Chris Welty, Guo Tong Xie and Elisa Kendall. Chapter 9, written by Uwe Assmann, Steffen Zschaler and

Gerd Wagner, deals with ontologies, metamodels and the model-driven paradigm. Chapter 10 will presents the use of ontologies in software development environments in the work of Káthia Marçal de Oliveira, Karina Villela, Ana Regina Rocha and Guilherme Horta Travassos. Finally, the topic of the last chapter of the book (Chap. 11) is a semantic upgrade and publication of legacy data by Jesús Barrasa Rodríguez.

As a complement to this book, the Alarcos Group (the research group of the editors) have created a web site (http://alarcos.inf-cr.uclm.es/ontoset) to store and share, in an open way and by using standardized formats, examples of interesting ontologies in the SET discipline. In addition to the examples referred to in the book, other examples of ontologies elaborated by the international community will be included in this web site.

Audience

The audience for this book is software engineering researchers and practitioners (professors, PhD and postgraduate students, industrial R&D departments, etc.). The reader is assumed to have previous knowledge of software engineering.

Acknowledgements

We would like to express our gratitude to all those individuals and parties who helped us produce this volume. In the first place, we would like to thank all the contributing authors and reviewers who helped improve the final version. Special thanks to Springer-Verlag and Ralf Gerstner, for believing in our project and for giving us the opportunity to publish this book. We would also like to thank José Carlos Villar Herrera and Tomás Martínez Ruíz of UCLM for their support during the developement of this book.

Finally, we would like to acknowledge the public organizations that financed this work, under the research projects CALIPO (TIC2003-07804-C05-03) and ENIGMAS (PBI-05-058).

Coral Calero
Francisco Ruiz
Mario Piattini

July 2006

Contents

1. Ontological Engineering: Principles, Methods, Tools and Languages

Oscar Corcho

Information Management Group, University of Manchester, Kilburn Building, Oxford Road M13 9PL. Manchester, United Kingdom, Oscar.Corcho@manchester.ac.uk,

Mariano Fernández-López

Escuela Politécnica Superior, Universidad San Pablo CEU, Ctra. de Boadilla del Monte km 5.300, 28668 Boadilla del Monte, Madrid, Spain, mfernandez.eps@ceu.es,

Asunción Gómez-Pérez

Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo s/n. 28660 Boadilla del Monte, Madrid, Spain, asun@fi.upm.es

1.1 Introduction

In 1991, the DARPA Knowledge Sharing Effort ([88], p. 37) envisioned a new way to build intelligent systems. It proposed the following:

> *Building knowledge-based systems today usually entails constructing new knowledge bases from scratch. It could be instead done by assembling reusable components. System developers would then only need to worry about creating the specialized knowledge and reasoners new to the specific task of their system. This new system would interoperate with existing systems, using them to perform some of its reasoning. In*

> *this way, declarative knowledge, problem-solving techniques and reasoning services would all be shared among systems. This approach would facilitate building bigger and better systems and cheaply.*

Static knowledge is modeled by means of ontologies while problem solving methods specify generic reasoning mechanisms. Both types of components can be viewed as complementary entities that can be used to configure new knowledge-based systems from existing reusable components.

Since DARPA's idea, considerable progress has been made in developing the conceptual bases to build technology that allows reusing and sharing knowledge components. Ontologies and problem solving methods (PSMs) have been created to share and reuse knowledge and reasoning behavior across domains and tasks. In this evolution, the most important fact has been the emergence of the Semantic Web. According to [10], the Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. This cooperation can be achieved by using shared knowledge components, and so ontologies and PSMs have become key instruments in developing the Semantic Web.

Currently, ontologies are widely used in knowledge engineering, artificial intelligence and computer science, in applications related to knowledge management, natural language processing, e-commerce, intelligent integration information, information retrieval, database design and integration, bio-informatics, education, etc.

In this chapter, we present the basics about ontologies, and show what activities should be carried out during the ontology development process, what principles should be followed in ontology design, and what methods, methodologies, software tools and languages are available to give support to each one of these activities. First, in Sect. 1.2, we define the word 'ontology' and we briefly explaining its roots in philosophy. Section 1.3 is devoted to explain which are the main components that can be used to model ontologies. In Sect. 1.4, we present the main ontology design principles. In Sect. 1.5, we describe the ontology development process in the context of the Semantic Web, where ontologies can be highly distributed and present many links among each other (hence the notion of networked ontologies). In Sect. 1.6, we describe the development of ontologies and the life cycle. In Sect. 1.7, we describe the methods, methodologies and tools commonly used for the whole ontology development process or only for specific activities. Among them we pay attention to those aimed at ontology learning, which reduce the effort needed during the knowledge ac-

quisition process; at ontology merging, which generates a unique target ontology from several source ontologies; at ontology alignment, which establishes different types of mappings between ontologies (hence preserving the original ones); and at ontology evaluation, which evaluates ontology content. In the implementation activity description, we present ontology languages that can be used to implement ontologies. Finally, conclusions and future lines of research are presented in Sect. 1.8.[1]

1.2 What Is an Ontology? Viewpoints from a Philosopher and from an Ontology Engineer

The ancient Greeks were concerned with the question: "what is the essence of things through the changes?" Many different answers to this question were proposed by Greek philosophers, from Parmenides of Elea (fifth and fourth centuries bc), the precursor of ontology, to Aristotle, author of the *MetaPhysics* (a work that might well have been called *Ontology*).

In his study of the essence of things, Aristotle distinguished different modes of being to establish a system of categories (*substance*, *quality*, *quantity*, *relation*, *action*, *passion*, *place* and *time*) to classify anything that may be *predicated* (said) about anything in the world. For example, when we say "this computer *is* on the table" we are assuming a different mode of being to when we say "this computer *is* gray". The first statement is classified inside the category of *place*, while the second is inside the category of *quality*. The categorization proposed by Aristotle was widely accepted until the eighteenth century.

In the modern age, Emmanuel Kant (1724–1804) provoked a *Copernican turn*. The essence of things is determined not only by the things themselves, but also by the contribution of whoever perceives and understands them. According to Kant, a key question is "what structures does our mind use to capture the reality?" The answer to this question leads to Kant's categorization. Kant's framework is organized into four classes, each of which presents a triadic pattern: *quantity* (*unity*, *plurality*, *totality*), *quality* (*reality*, *negation*, *limitation*), *relation* (*inherence*, *causality*, *community*) and *modality* (*possibility*, *existence*, *necessity*). Therefore, our mind classifies the object `John` as unique, real, existing, etc.

[1] For a deep introduction to the ontological engineering field, we recommend Gómez-Pérez and colleages' book [40].

A classification of categories, such as the ones mentioned above, is known as an ontology by philosophers [47]. Most modern examples of ontologies (in the context of philosophy) are due to Chisholm [16], Johanson [59], and Hoffman and Rosenkrantz [110], among others.

According to what we have said, it is very important to take into account that 'an ontology' is not the same as 'ontology'.An ontology is a classification of categories, whereas ontology is a branch of philosophy.

To answer our second question ("what is an ontology for an ontology engineer?"), we can assume that there is a parallelism between the reality perceived by people and by computers, and both can be structured in ontologies [44]. In accordance with this idea, if a computer is exclusively devoted to answering questions on travel, its reality could be structured by classifying travel as travel by train, travel by plane, etc. However, for this classification to be really an ontology for the computer, the computer must be able to reason with it. This leads to the first important difference between an ontology from a philosophical point of view and from a computer science point of view. According to the latter, an ontology has to be codified in a machine interpretable language [106, 39]. In other words, when an ontology engineer defines what an ontology is, (s)he changes the perspective from the person to the computer. Thus, if the computer does not 'understand' the ontology, it cannot be its ontology. Moreover, from a computer science point of view, an ontology is usually (although not necessarily) more specific than an ontology from a philosophical approach. Finally, due to the use of the term 'ontology', the features of reusability and shareability have become essential in the definition of this term for engineers. Nevertheless, such features are not essential in philosophical ontologies.

In conclusion, **for an ontology engineer** ([106], p. 185, with our own emphasis):

> ***An ontology*** *is a formal, explicit specification of a shared conceptualization. Conceptualization refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. Explicit means that the type of concepts used, and the constraints on their use, are explicitly defined. Formal refers to the fact that the ontology should be machine-readable. Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group.*

Neches and colleagues ([88], p. 40, our emphasis) gave another definition, focused on the form of an ontology:

> ***An ontology*** *defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary.*

1.3 What Are the Main Components of an Ontology?

Different knowledge representation formalisms (and corresponding languages) exist for the fomalization (and implementation) of ontologies. Each of them provides different components that can be used for these tasks. However, they share the following minimal set of components.[2]

Classes represent concepts, which are taken in a broad sense. For instance, in the traveling domain, concepts are: locations (cities, villages, etc.), lodgings (hotels, camping, etc.) and means of transport (planes, trains, cars, ferries, motorbikes and ships). Classes in the ontology are usually organized in taxonomies through which inheritance mechanisms can be applied. We can represent a taxonomy of entertainment places (theater, cinema, concert, etc.) or travel packages (economy travel, business travel, etc.). In the frame-based knowledge representation paradigm, metaclasses can also be defined. Metaclasses are classes whose instances are classes. They usually allow for gradations of meaning, since they establish different layers of classes in the ontology where they are defined.

Relations represent a type of association between concepts of the domain. They are formally defined as any subset of a product of n sets, that is: $R \subset C1 \times C2 \times \ldots \times Cn$. Ontologies usually contain binary relations. The first argument is known as the domain of the relation, and the second argument is the range. For instance, the binary relation `arrivalPlace` has the concept `Travel` as its domain and the concept `Location` as its range. Relations can be instantiated with knowledge from the domain. For example, to express that the flight AA7462-Feb-08-2002 arrives in Seattle we must write: `(arrivalPlace AA7462-Feb-08-2002 Seattle)`.

Binary relations are sometimes used to express concept attributes (aka slots). Attributes are usually distinguished from relations because their range is a datatype, such as *string*, *number*, etc., while the range of relations is a concept. The following code defines the attribute `flightNum-`

[2] Component names depend on the formalism. For example, classes are also known as concepts, entities and sets; relations are also known as roles and properties; etc.

ber, which is a *string*. We can also express relations of higher arity, such as "a road connects two different cities".

According to Gruber [44], ***formal axioms*** serve to model sentences that are always true. They are normally used to represent knowledge that cannot be formally defined by the other components. In addition, formal axioms are used to verify the consistency of the ontology itself or the consistency of the knowledge stored in a knowledge base. Formal axioms are very useful for infering new knowledge. An axiom in the traveling domain would be that it is not possible to travel from North America to Europe by train.

Instances are used to represent elements or individuals in an ontology. An example of an instance of the concept AA7462 is the flight AA7462 that arrives at Seattle on February 8, 2006 and costs 300 (US dollars, euros, or any other currency).

1.4 Ontological Engineering

The ontological engineering field has been subject to considerable study and research during the last decade. **Ontological engineering** refers to the set of activities that concern the ontology development process, the ontology life cycle, the principles, methods and methodologies for building ontologies, and the tool suites and languages that support them [39]. The notion of networked ontological engineering has come into play with the emergence of the Semantic Web, where one of the most relevant assumptions is that ontologies are distributed across different Web servers and ontology repositories and may have overlapping representations of the same or different domains.

With regard to **methods and methodologies**, several proposals have been reported for developing ontologies. In 1990, Lenat and Guha [72] published the general steps and some interesting points about the Cyc development. Some years later, in 1995, on the basis of the experience gained in developing the Enterprise Ontology [113] and the TOVE (TOronto Virtual Enterprise) project ontology [46] (both in the domain of enterprise modeling), the first guidelines were proposed and later refined in [111, 112]. At the 12th European Conference for Artificial Intelligence (ECAI'96), Bernaras and colleagues [9] presented a method used to build an ontology in the domain of electrical networks as part of the Esprit KACTUS [100] project. The methodology METHONTOLOGY [40] appeared at the same time and was extended in later papers [31, 32]. It was proposed for ontology construction by the Foundation for Intelligent

Physical Agents (FIPA),[3] which promotes interoperability across agent-based applications. In 1997, a new method was proposed for building ontologies based on the SENSUS ontology [109]. Some years later, the On-To-Knowledge methodology appeared as a result of the project with the same name [102]. A comparative and detailed study of these methods and methodologies can be found in [29].

All the previous methods and methodologies were proposed for building ontologies. However, many other methods have been proposed for specific tasks in the ontology development process, such as ontology re-engineering [42], ontology learning [3, 65], ontology evaluation [35, 40, 36, 38, 60, 61, 114, 50, 48], ontology evolution [67, 68, 92, 96, 97, 93, 104], ontology alignment [8, 14, 76, 95, 80, 101, 25, 26, 98], and ontology merging [103, 33, 107, 94], among others.

Ontology tools appeared later, in the mid-1990s, and can be classified in the following two groups:[4]

- Tools whose knowledge model maps directly to an ontology language, hence developed as ontology editors for that specific language. This group includes: the Ontolingua Server [27], which supports ontology construction with Ontolingua and KIF; OntoSaurus [109] with Loom; WebOnto [24] with OCML; OilEd [7] with OIL first, later with DAML+OIL, and finally with OWL; and SWOOP [62] and KAON2 [56] with OWL.
- Integrated tool suites whose main characteristic is that they have an extensible architecture, and whose knowledge model is usually independent of ontology languages. These tools provide a core set of ontology-related services and are easily extended with other modules to provide more functions. In this group we have included Protégé [91], WebODE [17, 1], OntoEdit [108], and KAON1 [77].

Ontology languages started to be created at the beginning of the 1990s, normally as the evolution of existing knowledge representation (KR) languages. Basically, the KR paradigms underlying such ontology languages were based on first order-logic (e.g., KIF [34]), on frames combined with first-order logic (e.g., Ontolingua [27, 43], OCML [87] and FLogic [66]), and on description logics (e.g., Loom [75]). In 1997, OKBC [15] was created as a unifying frame-based protocol to access ontologies implemented

[3] http://www.fipa.org/specs/fipa00086/ (last accessed, August 9, 2005).

[4] In each group, we have followed a chronological order of appearance in the enumeration of the tool.

in different languages (Ontolingua, Loom and CycL, among others). However, it was only used in a small number of applications.

The boom of the Internet led to the creation of ontology languages for exploiting the characteristics of the Web. Such languages are usually called *Web-based ontology languages* or *ontology markup languages*. Their syntax is based on existing markup languages such as HTML [99] and XML [12], whose purpose is not ontology development but data presentation and data exchange respectively. The most important examples of these markup languages are: SHOE [74], XOL [63], RDF [70], RDF Schema [13], OIL [54], DAML+OIL [55] and OWL [20]. From all of them, the ones that are being actively supported now are RDF, RDF Schema and OWL.

1.5 Principles for the Design of Ontologies

This section summarizes some design criteria and a set of principles that have been proven useful in the development of ontologies. According to [45], ontology design principles are objective criteria for guiding and evaluating ontology designs. He identified the following five principles:

- **Clarity** [45], which is defined in the following terms: *An ontology should communicate effectively the intended meaning of defined terms. Definitions should be objective. Definitions can be stated on formal axioms, and a complete definition (defined by necessary and sufficient conditions) is preferred over a partial definition (defined by only necessary or sufficient conditions). All definitions should be documented with natural language.*
- **Minimal encoding bias** [45], which means that: *The conceptualization should be specified at the knowledge level without depending on a particular symbol-level encoding.*
  Encoding bias should be minimized for knowledge sharing because agents that share knowledge may be implemented in different ways.
- **Extendibility** [45], which says that: *One should be able to define new terms for special uses based on the existing vocabulary, in a way that does not require the revision of the existing definitions.*
- **Coherence** [45], which is defined as follows: *An ontology should be coherent: that is, it should sanction inferences that are consistent with the definitions. [...] If a sentence that can be inferred from the axioms contradicts a definition or example given informally, then the ontology is incoherent.*

- **Minimal ontological commitments** [45], which is described in this way: *Since ontological commitment is based on the consistent use of the vocabulary, ontological commitment can be minimized by specifying the weakest theory and defining only those terms that are essential to the communication of knowledge consistent with the theory.*

According to this last principle, we should not commit to a specific format for dates, for currencies, etc., when designing our ontologies, since such details could be different in different systems.

Some other criteria have proven useful in ontology design, such as **the standardization of names** [2], which proposes to use the same of conventions to name related terms, in order to ease the understanding of the ontology.

1.6 Ontology Development Process and Life Cycle

In 1997, the ontology development process [31] was identified in the framework of the METHONTOLOGY methodology for ontology construction. Such a proposal was based on the IEEE standard for software development [57]. The ontology development process refers to the activities that have to be performed when building ontologies. They can be classified in the three categories presented in Fig. 1.1.

Ontology management activities include scheduling, control and quality assurance. The *scheduling* activity identifies the tasks to be performed, their arrangement, and the time and resources needed for their completion. This activity is essential for ontologies that use ontologies stored in ontology libraries or for ontologies that require a high level of abstraction and generality. The *control* activity guarantees that scheduled tasks are completed in the manner intended to be performed. Finally, the *quality assurance* activity assures that the quality of each and every product output (ontology, software and documentation) is satisfactory.

Ontology development-oriented activities are grouped, as presented in Fig. 1.1, into pre-development, development and post-development activities. During the pre-development, an *environment study* identifies the problem to be solved with the ontology, the applications where the ontology will be integrated, etc. Also during the pre-development, the *feasibility study* answers questions like: "is it possible to build the ontology?"; "is it suitable to build the ontology?"; etc.

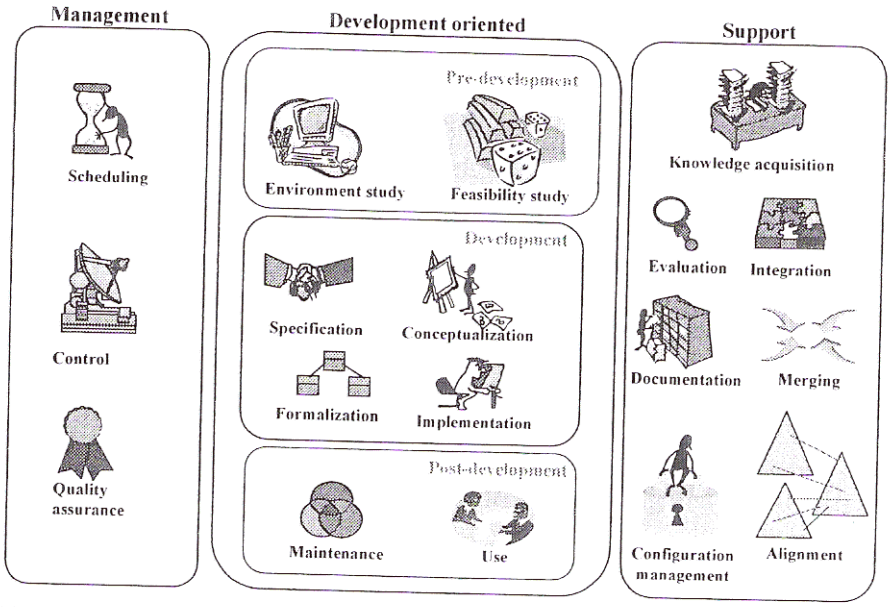

Fig. 1.1. Ontology development process (adapted from [31])

Once in development, the *specification* activity[5] states why the ontology is being built, what its intended uses are and who the end-users are. The *conceptualization* activity structures the domain knowledge as meaningful models at the knowledge level [89] either from scratch or by reusing existing models. In this last case, related activities are pruning branches of the existing taxonomies, extending the coverage of ontologies with the addition of new concepts in the higher levels of their taxonomies, or specializing branches that require more granularity. Given that the conceptualization activity is implementation-language independent, it allows modeling ontologies according to the minimal encoding bias design criterion. The *formalization* activity transforms the conceptual model into a formal or semi-computable model. The *implementation* activity builds computable models in an ontology language.

During post-development, the *maintenance* activity updates and corrects the ontology if needed. Also during post-development, the ontology

[5] In [28] *specification* is considered as a pre-development activity. However, following more strictly the IEEE standard for software development, the specification activity was considered part of the proper development process. In fact, the result of this activity is an ontology description (usually in natural language) that will be transformed into a conceptual model by the *conceptualization* activity.

is *(re)used* by other ontologies or applications. The evolution activity consists of managing ontology changes and their effects by creating and maintaining different variants of the ontology, taking into account that they can be used in different ontologies and applications [93].

Finally, **ontology support activities** include a series of activities that can be performed during the development-oriented activities, without which the ontology could not be built. They include knowledge acquisition, evaluation, integration, merging, alignment, documentation and configuration management. The goal of the *knowledge acquisition* activity is to acquire knowledge from experts in a given domain or through some kind of (semi-)automatic process, which is called ontology learning [65]. The *evaluation* activity [35] makes a technical judgment of the ontologies, of their associated software environments, and of the documentation. This judgment is made with respect to a frame of reference during each stage and between stages of the ontology's life cycle. The *integration* activity is required when building a new ontology by reusing other ontologies already available. Another support activity is *merging* [103, 33, 107, 94], which consists of obtaining a new ontology starting from several ontologies in the same domain. The resulting ontology is able to unify concepts, terminology, definitions, constraints, etc., from all the source ontologies. The merging of two or more ontologies can be carried out either in runtime or design time. The *alignment* activity establishes different kinds of mappings (or links) between the involved ontologies. Hence this option preserves the original ontologies and does not merge them. The *documentation* activity details, clearly and exhaustively, each and every one of the completed stages and products generated. The *configuration management* activity records all the versions of the documentation and of the ontology code to control the changes. The *multilingualism activity* consists of mapping ontologies onto formal descriptions of linguistic knowledge [22]. It has not usually been considered as an ontology support activity, but has become more relevant in the context of networked ontologies available in the Semantic Web.

As we can see, the ontology development process does not identify the order in which the activities should be performed [31] (see also [57]). This is the role of the **ontology life cycle**, which identifies *when* the activities should be carried out; that is, it identifies the *set of stages* through which the ontology moves during its lifetime, describes what activities are to be performed in each stage and how the stages are related (relation of precedence, return, etc.).

The initial version of the life cycle process model of METHONTOLOGY (see Fig. 1.2) proposes to start with a scheduling of the activities to be performed. Then, the specification activity begins,

showing why the ontology will be built, what its possible uses will be, and who are its users. When the specification finishes, the conceptualization begins. The objective of the conceptualization is to organize and structure the acquired knowledge in the knowledge acquisition activity, using a set of representations easy to manipulate for the experts in the domain. Once the conceptual model has been built, METHONTOLOGY proposes to automatically implement the ontologies using translators. More details can be found in [39].

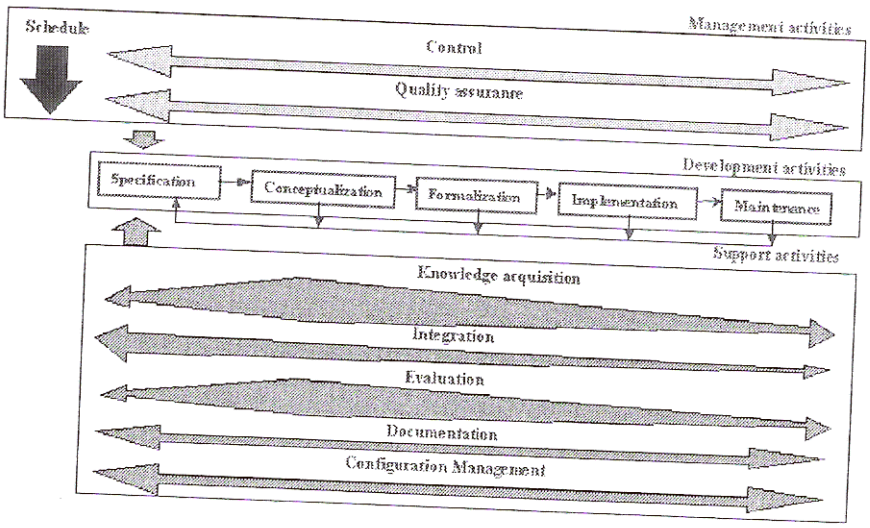

Fig. 1.2. Ontology life cycle in METHONTOLOGY

As more ontologies become available in ontology libraries or spread over the Internet, their reuse by other ontologies and applications increases. Domain ontologies can be reused to build others of more granularity and coverage, or can be merged with others to create new ones. Figure 1.3 shows different ways or possibilities of construction. Using an analogy with an underground map, it can be noted that there exists a main line (in the middle of the figure), others that start from the main line or finish in it, or lines that run parallel and fork in a point. Thus, *interdependence relationships* [42] arise between the life cycles of several ontologies, and actions of evaluation, pruning and merging can be carried out on such ontologies. That is, the life cycles of the different ontologies intersect, producing different scenarios with different technological requirements. Now we will describe some of the most common scenarios that arise during the ontology development process.

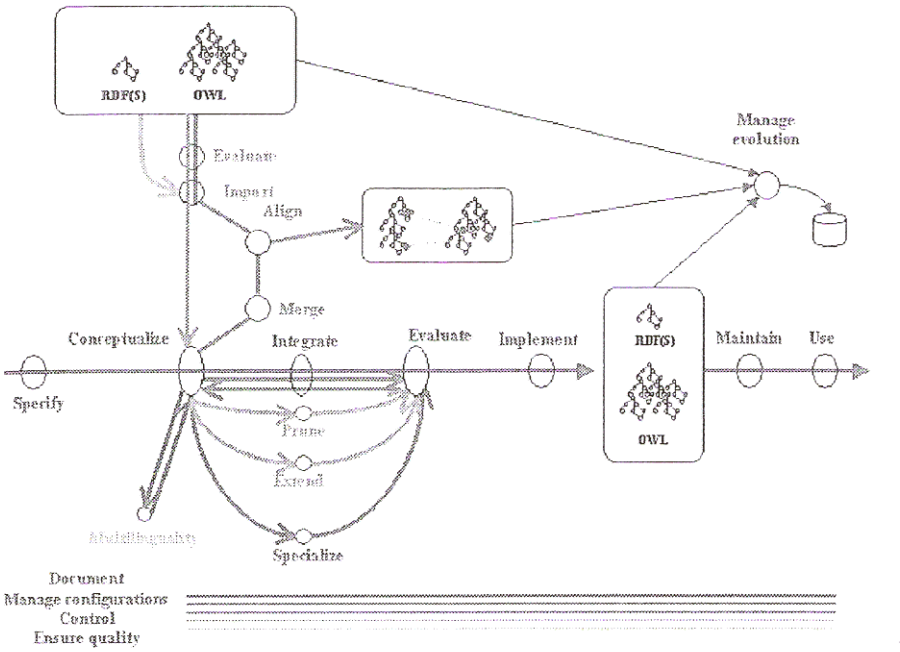

Fig. 1.3. The ontology development process of networked ontologies

- Scenario 1 **Evaluate + import**. The import of ontologies consists of incorporating an ontology available in a language or tool into another ontology tool. Often, several candidate ontologies implemented in different languages can be reused. In this case, it is necessary to inspect their content and granularity, compare them and select the best one(s). It is also necessary to analyze the expressiveness of the language in which each ontology is implemented, since important pieces of knowledge may be lost during the import if the knowledge model of the target ontology tool is less expressive than that of the language or tool where the ontology is implemented.

  Before importing an ontology, its content should be evaluated. Some ontology tools perform content evaluation before the import process, so as to avoid importing and reusing badly designed ontologies.

  Ontology import is not always successful. We can find problems due to lack of interoperability between tools, which are as follows [18]:

  – Common interchange formats normally allow representation of the same knowledge in many different ways and many of them have ex-

tensible knowledge models (by means of metaclasses). Hence, translations to and from interlinguas are usually written with regard to a specific format, making knowledge exchange difficult. Some of the interlinguas that have been used in the past are KIF and RDF.
- Interoperability with interlinguas has been only proved with the same origin and target formats, but not between different source and target formats.

However, with the standardization of the OWL language in the context of the Semantic Web the issue of importing ontologies has become less relevant in the ontology development process, since the OWL specification clearly states which primitives are allowed in OWL ontologies and how they can be combined. The use of OWL Full, which provides more expressivity possibilities, may again pose the same type of interoperability problems.

- Scenario 2. **Conceptualize + integrate + evaluate conceptualization**. Once an ontology has been imported, the next step consists of integrating its conceptual model into the conceptual model of the ontology that is being developed. Consequently, activities of integration and evaluation of the conceptualization are in the main line of the life cycle process model.
- Scenario 3. **Conceptualize + acquire knowledge**. Once the ontology has been evaluated, imported and integrated into the conceptual model of the main ontology, the activity of conceptual evaluation can reveal, both in the integrated ontologies and in the main ontology that is being developed, what parts of the ontologies are in the requirement specification document. Therefore, the options are:

  - To prune the branches of the taxonomy that are not considered necessary because they do not appear in the ontology requirement specification document.
  - To specialize those branches that require more granularity, including more specialized domain concepts and relations.
  - To extend the ontologies including (in width) new concepts and relations.
  - To search for new domain ontologies that complement the detected lacks.

If the ontology builder prunes, specializes or extends the ontologies, (s)he might need some knowledge that could be acquired using classical knowl-

edge engineering methods and techniques, or semi-automatic methods for learning ontologies from texts or other sources.

- Scenario 4. **Semi-automatic construction of ontologies**. Despite one of the main objectives of ontologies being to decrease the knowledge acquisition bottleneck when building knowledge-intensive systems, the process of building an ontology and refining it consumes much time and resources. *Ontology learning* is the process that partially automates the construction of ontologies using some of the following methods, techniques and tools: natural language analysis, statistical methods, linguistic patterns, text mining, etc. This process uses texts, electronic dictionaries, linguistic ontologies (like WordNet), and structured and semi-structured information and data as knowledge sources.
- Scenario 5. **Evaluate and import a set of ontologies, and align them**. Quite often there are several ontologies that model the same domain. There can be situations in which we want to compare ontologies in the same domain to determine what terms of an ontology map terms of another. The correspondences between ontologies obtained by means of this procedure are called mappings. In this case, the result of this scenario is a set of mappings that establish the relationships between the two domain ontologies. There are also situations where the relationships are established between ontologies of different categories, as in the case of joining a domain ontology with an upper level ontology. Every alignment requires the evaluation and possibly the import of the ontologies in specific tools, and the generation of the result in a language where the content of the alignment can be evaluated.
- Scenario 6. **Evaluate and import a set of ontologies, and merge them**. This scenario is an extension of scenario 5. Once the mappings between the ontologies are known, the ontology engineer can merge them in a new ontology.
- Scenario 7. **Translate the ontology into another natural language (Spanish, English, French, etc.)**. Once the ontology has been conceptualized, it can require the translation of all its terms into another language using multilingual thesauri and electronic dictionaries (e.g., EuroWordNet).
- Scenario 8. **Manage the evolution of the ontology**. Given that an essential feature of ontologies is that they must be agreed, the most natural way of developing ontologies is through a collaborative construction. Ontology engineers working in parallel on the same ontology need to maintain and compare different versions, to examine the changes that others have performed, and to accept or reject the

changes [93]. In a networked environment like the Semantic Web, we should keep control of the versions of each ontology and the relationships between them must be kept for several reasons. One of the reasons is that semantic markup annotations of Web resources are based on ontologies. If an ontology is changed, we should be able to deal with the effects of the change in the existing annotations. Another reason is that ontologies will be developed in different tools and languages. We must be able to keep control of their life cycle when they are exchanged back and forward between different ontology tools and when they are translated from and to different ontology languages.

- Scenario 9. **Perform support activities**. The activities of documentation, evolution management, configuration management, and quality assurance and control are carried out during the whole development process.

1.7 Methods, Methodologies, Tools and Languages

In this section, first of all we explain the methods, methodologies and tools used for the whole ontology development process. Then we focus on ontology learning, ontology merging, ontology alignment, ontology evolution and versioning, ontology evaluation and ontology implementation (in this last section we review existing ontology languages).

1.7.1 Methods, Methodologies and Tools Used for the Whole Ontology Development Life Cycle

This section presents the classical methodologies and methods used to build ontologies from scratch or by reusing other ontologies, and the new generation of ontology building platforms that support the ontology development process.

1.7.1.1 Methodologies and Methods

Concerning methods and methodologies, the approaches dealt with are the Cyc method, the Uschold and King method, the Grüninger and Fox methodology, the KACTUS approach, METHONTOLOGY, the SENSUS method, and the On-To-Knowledge methodology.

The method used to build the Cyc knowledge base [72] consists of three phases. The first phase consists of the manual coding of articles and pieces of knowledge, in which common sense knowledge that is implicit in dif-

ferent sources is extracted by hand. The second and third phases consist of acquiring new common sense knowledge using natural language or machine learning tools. The difference between them is that in the second phase this common sense knowledge acquisition is aided by tools, but mainly performed by humans, while in the third phase the acquisition is mainly performed by tools. This approach is applicable, besides the main line of the life cycle model, also to scenario 4 (semi-automatic construction of ontologies). According to the Cyc method, the resulting ontology will be divided into microtheories (or contexts), bundles of assertions in the same domain, and is implemented in the CycL language.

The **Uschold and King** method [113] proposes four phases: (1) to identify the purpose of the ontology, (2) to build it, (3) to evaluate it, and (4) to document it. During the building phase, the authors propose capturing knowledge, coding it and integrating other ontologies inside the current one. The authors also propose three strategies for identifying the main concepts in the ontology: a top-down approach, in which the most abstract concepts are identified first and then specialized into more specific concepts; a bottom-up approach, in which the most specific concepts are identified first and then generalized into more abstract concepts; and a middle-out approach, in which the most important concepts are identified first and then generalized and specialized into other concepts. This approach is applicable, besides the main line of the life cycle model, also to scenario 1 (evaluate and import).

Grüninger and Fox [46] propose a methodology that is inspired by the development of knowledge-based systems using first-order logic. They propose first to identify intuitively the main scenarios (possible applications in which the ontology will be used). Then, a set of natural language questions, called competency questions, are used to determine the scope of the ontology. These questions and their answers are used to extract the main concepts and their properties, relations and axioms of the ontology. Such ontology components are formally expressed in first-order logic. Therefore, this is a very formal methodology that takes advantage of the robustness of classical logic. It can be used as a guide to transform informal scenarios in computable models. This approach is applicable, besides the main line of the life cycle model, also to scenario 1 (evaluate + import) and scenario 2 (conceptualize + integrate + evaluate conceptualization).

In the method proposed in the **KACTUS** project [9] the ontology is built on the basis of an application knowledge base (KB), by means of a process of abstraction (i.e., following a bottom-up strategy). The more the applications are built, the more general the ontology becomes; hence, the further the ontology moves away from a KB. In other words, the authors propose to start building a KB for a specific application. Later, when a new KB in a

similar domain is needed, they propose to generalize the first KB into an ontology and adapt it for both applications. Applying this method recursively, the ontology would represent the consensual knowledge needed in all the applications. A way to apply this approach is to generate an ontology by means of the generalization of several KBs that model the same domain. Therefore, the KACTUS method is applicable, besides the main line of the ontology life cycle model, also to scenario 6 (evaluate and import *N* ontologies (or KBs to merge them).

The method based on **Sensus** [109] is a top-down approach for deriving domain-specific ontologies from huge ontologies. The authors propose to identify a set of "seed" terms that are relevant to a particular domain. These terms are linked manually to a broad-coverage ontology (in this case, the Sensus ontology, which contains more than 50,000 concepts). Then, all the concepts in the path from the seed terms to the root of Sensus are included. If a term that could be relevant in the domain has not yet appeared, it is added manually, and the previous step is performed again, until no term is missing. Finally, for those nodes that have a large number of paths through them, the entire subtree under the node is sometimes added, based on the idea that if many of the nodes in a subtree have been found to be relevant, then the other nodes in the subtree are likely to be relevant as well. Consequently, this approach promotes the sharebility of knowledge, since the same base ontology is used to develop ontologies in particular domains. This method is especially useful in scenario 3 (conceptualize + acquire knowledge).

METHONTOLOGY [32] is a methodology, created by the Ontological Engineering Group of the Technical University of Madrid (UPM), for building ontologies either from scratch, reusing other ontologies as they are, or by a process of reengineering them. The METHONTOLOGY framework enables the construction of ontologies at the knowledge level. It includes: the identification of the ontology development process, a life cycle based on evolving prototypes (based in the one presented in Figs. 1.2 and 1.3), and particular techniques to carry out each activity. The main phase in the ontology development process using the METHONTOLOGY approach is the conceptualization phase. METHONTOLOGY considers all the scenarios presented in Sect. 1.6. In fact, such scenarios were born as a consequence of the methodological studies at UPM in the MKBEEM (IST-1999-10589) and Esperonto (IST-2001-34373) projects.

The **On-To-Knowledge** methodology [102] includes the identification of goals that should be achieved by knowledge management tools and is based on an analysis of usage scenarios. The steps proposed by the methodology are: *kick-off*, where ontology requirements are captured and specified, competency questions are identified, potentially reusable ontologies

are studied and a first draft version of the ontology is built; *refinement*, where a mature and application-oriented ontology is produced; *evaluation*, where the requirements and competency questions are checked, and the ontology is tested in the application environment; and *ontology maintenance*. With regards to the scenarios to which this methodology is applicable, its authors are researching activities concerning all the scenarios; however, an ontology life cycle model that guides the order to carry out the different activities for the scenarios alternative to the main line has not been proposed. Besides, some scenarios are not integrally considered, e.g., scenario 8 (documentation + evolution management + configuration management + quality assurance and control).

If we analyze the approaches according to the part of the ontology development process that they describe, we can conclude that (see [29]):

- None of the approaches covers all the processes involved in ontology building. Most of the methods and methodologies for building ontologies are focused on the development activities, especially on ontology conceptualization and ontology implementation, and they do not pay too much attention to other important aspects related to management, learning, merging, integration, evolution and evaluation of ontologies. Therefore, such methods should be added to the methodologies for ontology construction from scratch (see an example in [30]).
- Most of the approaches are focused on development activities, especially ontology implementation, and they do not pay too much attention to other important aspects related to the management, evolution and evaluation of ontologies. This is due to the fact that the ontological engineering field is relatively new. However, a low compliance with the criteria formerly established does not mean a low quality of the methodology or method. As [53] states, a not very specified method can be very useful for an experienced group.
- Most of the approaches present some drawbacks in their use. Some of them have not been used by external groups and, in some cases, they have been used in a single domain.
- Most of the approaches do not have a specific tool that gives them technological support. Besides, none of the available tools covers all the activities necessary in ontology building.

1.7.1.2 Ontology Tools

Concerning the software platforms that give support to most of the activities of the ontology development life cycle, we will focus on the new gen-

eration of ontology engineering environments, in particular, on Protégé, WebODE, OntoEdit and KAON1.[6] They have been created to integrate ontology technology in actual information systems. As a matter of fact, they are built as robust integrated environments or suites that provide technological support to most of the ontology life cycle activities. They have extensible, component-based architectures, where new modules can easily be added to provide more functionality to the environment. Besides, the knowledge models underlying these environments are language independent.

Protégé [91] has been developed by Stanford Medical Informatics (SMI) at Stanford University. It is an open source, standalone application with an extensible architecture. The core of this environment is the ontology editor, and it holds a library of plugins that add more functionality to the environment. Currently, plugins are available for ontology language import/export (FLogic, Jess, XML, Prolog), ontology language design [69], OKBC access, constraints creation and execution (PAL), ontology merge (Prompt [94]), etc. This platform provides support for the main line of the life cycle model, for scenario 1 (evaluate and import), scenario 2 (conceptualize + integrate + evaluate conceptualization), scenario 3 (conceptualize + acquire knowledge), scenario 5 (evaluate and import *N* ontologies to align them), scenario 6 (evaluate and import *N* ontologies to merge them) and scenario 8 (manage the evolution of the ontology).

WebODE [17, 1] is the successor of ODE (Ontology Design Environment) [11], and has been developed at UPM. It is also an ontology engineering suite created with an extensible architecture. WebODE is not used as a standalone application, but as a Web server with several frontends. The core of this environment is the ontology access service, which is used by all the services and applications plugged into the server, especially by the WebODE Ontology Editor. There are several services for ontology language import/export (XML, RDF(S), OWL, CARIN, FLogic, Jess, Prolog), axiom editing, ontology documentation, ontology evaluation and ontology merging. WebODE's ontologies are stored in a relational database. Finally, WebODE covers and gives support to most of the activities involved in the ontology development process proposed by METHONTOLOGY, although this does not prevent it from being used with other methodologies or without following any methodology. This platform also provides support for the main line of the life cycle model and for scenarios 1, 2, 3, 6 and 7 (translate the ontology into another natural language).

[6] Other tools (Ontolingua Server, OntoSaurus, WebOnto, etc.) are described in [40].

OntoEdit [108] has been developed by AIFB at Karlsruhe University, and commercialized by Ontoprise. It is similar to the previous tools: it is an extensible and flexible environment, based on a plugin architecture, which provides functionality to browse and edit ontologies. It includes plugins that are in charge of inferring using Ontobroker, of exporting and importing ontologies in different formats (FLogic, XML, RDF(S) and OWL), etc. Two versions of OntoEdit are available: OntoEdit Free and OntoEdit Professional. This platform provides support for the main line of the life cycle model, for scenario 1 (evaluate and import) and scenario 2 (conceptualize + integrate + evaluate conceptualization).

The **KAON1** tool suite [77] is an open source extensible ontology engineering environment. The core of this tool suite is the ontology API, which defines its underlying knowledge model based on an extension of RDF(S). The OI modeler is the ontology editor of the tool suite that provides capabilities for ontology evolution, ontology mapping, ontology generation from databases, etc. This platform provides support for the main line of the life cycle model, for scenario 1 (evaluate and import), scenario 2 (conceptualize + integrate + evaluate conceptualization), scenario 3 (conceptualize + acquire knowledge), scenario 4 (semi-automatic construction of ontologies), scenario 6 (evaluate and import N ontologies to merge them) and scenario 8 (evolve the ontology).

An interesting aspect of tools is that only OntoEdit and WebODE give **support to ontology building methodologies** (On-To-Knowledge and METHONTOLOGY respectively), though this does not prevent them from being used with other methodologies or with no methodology at all.

From the **KR paradigm** point of view, KAON is based on semantic networks plus frames, and the rest of the tools allow the representation of knowledge following a hybrid approach based on frames and first-order logic. **Expressiveness of the underlying tool knowledge model** is also important. All the tools allow the representation of classes, relations, attributes and instances. Only KAON1, and Protégé provide flexible modeling components like metaclasses. Before selecting a tool for developing an ontology, it is also important to know the **inference services** attached to the tool, which include: constraint and consistency checking mechanisms, type of inheritance (single, multiple, monotonic, non-monotonic), automatic classifications, exception handling and execution of procedures. KAON1 does not have an inference engine. OntoEdit uses FLogic [66] as its inference engine, WebODE uses Ciao Prolog [52], and Protégé uses an internal PAL engine. Further, Protégé and WebODE provide ontology evaluation facilities and also include a module that performs ontology evaluation according to the OntoClean method [114, 50]. Finally, Protégé

(with the OWL plugin) performs automatic classifications by means of connecting to a description logic reasoner.

Another important aspect to take into account in ontology tools is the **software architecture and tool evolution**, which considers which hardware and software platforms are necessary to use the tool, its architecture (standalone, client/server, *n*-tier application), extensibility, storage of ontologies (databases, ASCII files, etc.), failure tolerance, backup management, stability and tool versioning policies. From that perspective, all these tools are based on Java platforms and provide database storage support. Backup management functionality is just provided by WebODE, and extensibility facilities are allowed in KAON, OntoEdit, Protégé and WebODE.

Interoperability with other ontology tools, information systems and databases, as well as translations to and from some ontology languages, is another important feature in order to integrate ontologies into applications. Most of the tools export and import to ad hoc XML and other ontology markup languages. However, there is no comparative study on the quality of all these translators. Moreover, there are no empirical results about the possibility of exchanging ontologies between different tools and about the amount of knowledge that is lost in the translation processes. Some effort in this regard has been carried out in the EON 2004 workshop.[7]

Related to the **cooperative and collaborative construction of ontologies**, Protégé incorporates some synchronization functionalities. In general, more features are required in existing tools to ensure a successful collaborative building of ontologies.

1.7.2 Ontology Learning

Ontology learning is defined as the set of methods and techniques used for building an ontology from scratch, enriching or adapting an existing ontology in a semi-automatic fashion using distributed and heterogeneous knowledge and information sources, allowing a reduction in the time and effort needed in the ontology development process. Though the fully automatic acquisition of knowledge remains far off, the overall process is considered as semi-automatic, meaning that human intervention is necessary in some parts of the learning process. Several approaches have appeared during the last decade for the partial automization of the knowledge acquisition process. To carry out this automization, natural language analysis and machine learning techniques can be used. This involves the

[7] http://km.aifb.uni-karlsruhe.de/ws/eon2004/

inclusion of a number of complementary disciplines that feed on different types of unstructured, semi-structured and fully structured data to support semi-automatic, cooperative ontology engineering [78].

Regarding ***ontology learning methods***, some of the best known ones are due to Maedche and colleagues [65], Aussenac-Gilles and colleagues [3, 4] and Khan and Luo [64]. **Maedche and colleagues' method** [65] proposes to learn the ontology using as a base a core ontology (Sensus, WordNet, etc.), which is enriched with the learned concepts. New concepts are identified using natural language analysis techniques over the resources previously identified by the user. The resulting ontology is pruned and then focused on a specific domain by means of several approaches based on statistics. Finally, relations between concepts are established applying learning methods. Such relations are added to the resulting ontology.

Aussenac-Gilles and colleagues' method [3, 4] is based on knowledge elicitation from technical documents. The method allows the creation of a domain model by analyzing a corpus with NLP tools. The method combines knowledge acquisition tools based on linguistics with modeling techniques to keep links between models and texts. After selecting a corpus, the method proposes to obtain linguistic knowledge (terms, lexical relations and groups of synonyms) at the linguistic level. This linguistic knowledge is then transformed into a semantic network. The semantic network includes concepts, relationships between concepts and attributes for the concepts.

Khan and Luo's method [64] aims to build a domain ontology from text documents using clustering techniques and WordNet [83]. The user provides a selection of documents regarding the same domain. Using these documents, a set of clusters where each cluster may contain more than one document is created, and then put into the correct place in a hierarchy. Each node in this hierarchy is a cluster of documents. For this purpose, the method proposes to use a modified algorithm, called the SOAT algorithm [115]. After building a hierarchy of clusters, a concept is assigned to each cluster in the hierarchy in a bottom-up fashion. First, concepts associated with documents are assigned to leaf nodes in the hierarchy. For each cluster of documents, these will be assigned a keyword-called topic that represents its content and uses predefined topic categories. For this purpose, a topic tracking algorithm [58] is used. Then, this topic is associated with an appropriate concept in WordNet. And finally, the interior node concepts is assigned according to the concepts in the descendant nodes and their hypernyms in WordNet. The type of relation between concepts in the hierarchy is ignored; it is only possible to know that there is a relation between them.

As we can see in this review, most of the ontology learning approaches are based on using linguistic patterns for extracting linguistic relations which would reflect ontological relations (taxonomic and non-taxonomic relations as well as possible attributes or their values, depending on the pattern's type). In the same sense, these kinds of patterns are also used for detecting attribute–value pairs. All the presented methods require the participation of an ontologist to evaluate the final ontology and the accuracy of the learning process. There are no methods or techniques for evaluating the accuracy of the learning process either.

With regard to ***ontology learning tools***, we focus on Caméléon [5], LTG Text Processing Workbench [82], Prométhée [85, 86], SOAT tool [115] and Text-To-Onto [78]. **Caméléon** [5] assists in learning conceptual relations to enrich conceptual models. Caméléon relies on linguistic principles for relation identification: lexico-syntactic patterns are good indicators of semantic relations. Some patterns may be regular enough to indicate the same kind of relation from one domain to another. Other patterns are domain specific and may reveal domain-specific relations. This tool gives technological support to some steps of the Aussenac-Gilles and colleagues' method.

LTG (Language Technology Group) **Text Processing Workbench** [82] is a set of computational tools for uncovering internal structure in natural language texts written in English. The main idea behind the workbench is the independence of the text representation and text analysis. In LTG, ontology learning is performed in two sequential steps: representation and analysis. At the representation step, the text is converted from a sequence of characters to features of interest by means of annotation tools. At the analysis step, those features are used by tools of statistics gathering and inference to find significant correlations in the texts. The workbench is being used both for lexicographic purposes and for statistical language modeling.

Prométhée [85, 86] is a machine learning-based tool for extracting and refining lexical–syntactic patterns related to conceptual specific relations from technical corpora. It uses pattern bases, which are enriched with the ones extracted in the learning. To refine patterns, the authors propose the Eagle [49] learning system. This system is based on the inductive paradigm *learning from examples*, which consists of the extraction of intentional descriptions of target concepts from their extensional descriptions, and previous knowledge of the given domain. This fact specifies general information, like the object characteristics and their relations. The tool extracts *intentional* descriptions of concepts from their *extensional* descriptions. The learned definitions are later used in recognition and classification tasks.

SOAT [115] allows semi-automatic domain ontology acquisition from a domain corpus. The main objective of the tool is to extract relationships from parsed sentences based on applying phraserules to identify keywords with strong semantic links like hypernyms or synonyms. The acquisition process integrates linguistic, common sense and domain knowledge. The restrictions of SOAT mean that the quality of the corpus must be very high, in the sense that the sentences must be accurate and sufficient to include most of the important relationships to be extracted.

Text-To-Onto [78] integrates an environment for building domain ontologies from an initial core ontology. It also discovers conceptual structures from different German sources using knowledge acquisition and machine learning techniques. Text-To-Onto has implemented some techniques for ontology learning from free and semi-structured text. The result of the learning process is a domain ontology that contains domain-specific and domain-independent concepts. Domain-independent concepts are withdrawn to better adjust the vocabulary of the domain ontology. The result of this process is a domain ontology that only contains domain concepts learned from the input sources related before. The ontologist supervises the whole process. This is a cyclic process, in the sense that it is possible to refine and complete the ontology if we repeat the process.

An important conclusion that we can obtain in the revision of ontology learning tools is that there is no fully automatic tool that carries out the learning process. Some tools are focused on helping in the acquisition of lexico-semantic knowledge, others help to elicit concepts or relations from a preprocessed corpus with the help of the user, etc. A deeper description of methods and tools can be found in [41].

1.7.3 Ontology Alignment and Merging

Ontologies aim to capture the consensual knowledge of a given domain in a generic and formal way, to be reused and shared across applications and by groups of people. From this definition we could wrongly infer that there is only one ontology for modeling each domain (or even a single universal ontology). Though this can be the case in specific domains, commonly several ontologies model the same domain knowledge in different ways. For instance, in the e-commerce field there are several standards and joint initiatives for the classification of products and services (UNSPSC,[8] e-cl@ss,[9] RosettaNet,[10] NAICS,[11] SCTG,[12] etc.). This hetero-

[8] http://www.unspsc.org/
[9] http://www.eclass.de/

geneity of ontologies also happens in many other domains (medicine, law, art, sciences, etc.).

Noy and Musen [94] defined ontology alignment and merging as follows: (1) *ontology alignment* consists of establishing different kinds of mappings (or links) between two ontologies, hence preserving the original ontologies (see Fig. 1.4); and (2) *ontology merging* proposes to generate a unique ontology from the original ontologies. In this chapter we will assume that a *mapping* between ontologies is a set of rewriting rules that associates terms and expressions defined in a source ontology with terms and expressions of a target ontology (inspired from [84]). Table 1.1 shows the mappings that can be established between the two ontologies of Fig. 1.4. The symbol := means is transformed into, and λ is the empty word. Therefore, `date := λ` means that the attribute date has no correspondence with terms of the ontology 2.

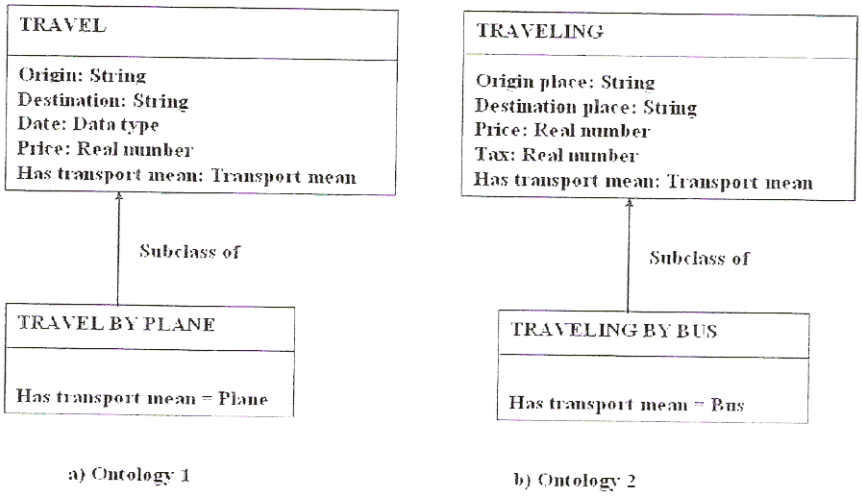

Fig. 1.4. Example of ontology alignment

Given that a reusable and machine interpretable database schema can be considered as an ontology (see Sect. 1.2), the galaxy of ***ontology alignment methods*** is huge. Some examples of these methods are: **S-Match** [101], **QOM** [25], **Pan and colleagues' proposal** [98], Artemis [8, 14], Cupid [76], AnchorPrompt [95], Similarity Flooding [80], etc. We will fo-

[10] http://www.rosettanet.org/
[11] http://www.naics.com/
[12] http://www.bts.gov/programs/cfs/sctg/welcome.htm

cus on the methods in bold because they show a complementary view of the problem of ontology alignment, although the three proposals share a lot of common ideas. Following Ehring and Staab's paper [25], we will present a process that subsumes most of these methods, and that allows us to comment them inside an integrated framework. The activities of this process are as follows (Ehring and Staab [25], inspired from CRISP-CM, the Cross Industry Standard Process for Data Mining)[13]:

- Activity 1. **Ontology adaptation**. The source ontologies are transformed into a format that is interpretable by the software that will carry out the process of alignment.

  QOM identifies this activity, but it does not explain how to carry it out. However, S-Match provides details on ontology adaptation. According to this method, names are morphologically analyzed in order to find all their possible basic forms (e.g., travels would be identified as a variation of travel).

  S-Match assumes that ontologies can be translated before aligning them. Further, it proposes that prepositions, conjunctions, etc., are transformed into logical connectives. For instance, `travel by plane` could be translated into "`C such as subclassOf(C, Travel) ∧ C.transporMean = Plane`".

  Concerning Pan and colleagues' proposal, their method transforms the two ontologies into a Bayesian network.
- Activity 2. Selection of the search space. If the ontologies have a large number of terms, pruning is necessary to avoid checking all the possible pairs of concepts.

  QOM gives guidelines to select the search space. Some of the identified strategies are:
  - Random. It limits the number of candidate mappings by selecting either a fixed number or a percentage from all possible mappings.
  - Label. It restricts candidate mappings to pairs of terms whose labels are near to each other in a sorted list. That is, each term could be possibly mapped with those that have a similar name.
  - Proximity. If two concepts are mapped, then it is very likely that some of their descendants or immediatelyrelated terms are also mapped.
  - Combination of different heuristics. The above-mentioned strategies can be combined to prune the candidate mappings.

  S-Match and Pan and colleagues' approach do not identify this activity.

[13] http://www.crisp-dm.org/

- Activity 3. **Similarity computation**. To choose the exact mappings between the possible pairs of terms, *similarity measures* are used. They associate likeness values to pairs of terms. The similarity measures compare term names, documents annotated with the terms, etc.
- Activity 4. **Similarity aggregation**. In general, there may be several similarity values for a candidate pair of terms from two ontologies, e.g., one for the similarity of their names and one for the similarity of their natural language descriptions. These different similarity values for one candidate pair must be aggregated into a single aggregated similarity value. A way to obtain the aggregated value is through the weighted means of the similarity values. This is basically QOM's approach. Neither S-Match nor Pal and colleagues' method combine similarities.
- Activity 5. **Interpretation**. Individual or aggregated similarity values are used to derive mappings between terms.

  QOM applies a threshold to discard spurious evidence of similarity, and considers as best mappings those with the highest aggregated similarity scores. Pan and colleagues follow a probabilistic and statistical analysis. Thus, if $p(C_1 \wedge C_2)$ is high, then C_1 and C_2 have a high overlap. If $p(C_1/C_2)$ is high and $p(C_2/C_1)$ is low, then it is very likely that C_1 is subclass of C_2. S-Math directly obtains the mappings trough the rules mentioned in activity 3.
- Activity 6. **Iteration**. Several methods perform an iteration over the whole process in order to take advantage of the knowledge already acquired in the first round. QOM iterates to find mappings based on lexical similarities first, and based on the structure of the ontologies later.

  S-Match proposes that the mappings generated during the first round are used as a KB to deduce other mappings. Thus, for instance, if we already know that the concept C_1 is equivalent to the concept C_2, and the concept C_2 is a subclass of the concept C_3, then we should be able to deduce that C_1 is a subclass of C_3.

  In the case of Pan and colleagues' proposal, there are mappings that can be deduced using the network obtained in the first round as a probabilistic KB.

Concerning ***ontology alignment tools***, the **QOM toolset** [25] gives support to the QOM method, presented in this section. It is implemented in Java using the KAON framework (see Sect. 1.7.1.2). It has been basically used to make experiments with the method and compare it with other methods.

Table 1.1. Mappings for the two ontologies of Fig. 1.4

Description of the mapping in natural language	Rewriting rule
The concept `travel` (in ontology 1) is equivalent to the concept `traveling` (in ontology 2).	Travel := Traveling
The concept `travel by plane` (in ontology 1) is equivalent to the concept such it is a subclass of `traveling` (in ontology 2) and its transport mean is a `plane` (in ontology 2).	TravelByPlane := C such as subclassOf(C, Traveling) ∧ C.hasTransporMean = Plane
The concept such it is a subclass of `travel` (in ontology 1) and its transport mean is a `bus` (in ontology 2) is equivalent to the concept `traveling by bus` (in ontology 2).	C such as subclassOf(C, Travel) ∧ C.hasTransporMean = Bus := TravelingByBus
The attribute `origin` (in ontology 1) is equivalent to the attribute `origin place` (in ontology 2).	Origin := OriginPlace
The attribute `destination` (in ontology 1) is equivalent to the attribute `destination place` (in ontology 2).	Destination := DestinationPlace
The value `New York` of attributes `origin` and `destination` (in ontology 2) is equivalent to the value `NY` of `origin place` and `destination place` (in ontology 2).	"New York" := "NY"
The attribute `date` (in ontology 1) does not have correspondence in ontology 2.	Date := λ
The attribute `price` (in ontology 1) is equivalent to a combination of the attributes `price` and `tax` in ontology 2.	Price := Price * (1 + Tax/100)
The attribute `has transport mean` (in ontology 1) is equivalent to the attribute `has transport mean` in ontology 2.	HasTransportMean := HasTransportMean

The **S-Match** tool translates and preprocesses the input ontologies. Then, it orders the transformation of prefixes, the expansions of abbreviations, etc. Later, using resources like WordNet, it generates a first mapping base. Finally, using the SAT solvers, new mappings are generated.

Pan and colleagues [98] apply their method by combining the Google search engine and text classifiers (such as Rainbow[14] or cbacl[15]) to calculate the prior probabilities of the Bayesian network. Then, the subsequent probability is calculated using any Bayesian network tool.

OLA[16] [26] is an API for manipulating alignments between ontologies in OWL. It allows applying and combining different algorithms, and even adding newones. Currently, this API has been mainly used with mapping methods based on lexical similarity measures. OLA implements a format for expressing alignments in RDF.

With regard to ***ontology merging methods and methodologies***, one of the most elaborated proposals for ontology merging is **ONIONS** [103, 33],

[14] http://www-2-cs-cmu.edu/~mccallum/bow/rainbow
[15] http://www.lbreyer.com/
[15] http://co4.inrialpes.fr/align

developed by the Conceptual Modeling Group of the CNR in Rome, Italy. With this method we can create a library of ontologies originating from different sources. The main underlying ideas of this method are: (1) to link the ontologies taking into account lexical relations between their terms (polysemy, synonymy, etc.); and (2) to use generic theories (part–whole or connectedness theories, for example) as common upper ontologies of the library ontologies; that is, to use generic theories as the glue to integrate the different ontologies.

FCA-Merge [107] was developed at the AIFB Institute of the University of Karlsruhe, Germany. This approach is very different from the other approaches presented in this section. FCA-Merge takes as input the two ontologies to be merged and a set of documents on the domains of the ontologies. The appearances of instances of the concepts in the different documents guides the merging of such concepts.

The **PROMPT** method [96] has been elaborated by the Stanford Medical Informatics Group at Stanford University. The main assumption of PROMPT is that the ontologies to be merged are formalized with a common knowledge model based on frames. This method proposes first to elaborate a list with the candidate operations to be performed to merge the two ontologies (e.g., merge two classes, merge two slots, etc.). Afterwards, a cyclic process starts. In each cycle the ontologist selects an operation of the list and executes it.

PromptDiff is a component of Prompt [97] that allows maintaining ontology views or mappings between ontologies. PromptDiff provides an ontologycomparison API that other applications can use to determine, for example, the mapping needs to be updated when new versions of mapped ontologies appear [93].

Concerning ***merging tools***, in the mid-1990s, research groups at the Universidad del País Vasco, MCC and the University of Georgia began to develop **OBSERVER** [81]. This tool automatically merged ontologies of the same domain to access heterogeneous information sources. However, the merge process was carried out by an internal module and, therefore, it was invisible to the user. Several years later, in the late 1990s, two groups at Stanford University developed two of the most relevant ontology merge tools: Chimaera and the Prompt plugin.

Chimaera [79] was built by the Knowledge Systems Laboratory (KSL) to aid in the process of ontology merge, and the **Prompt plugin** [94], integrated in Protégé, was built by the Stanford Medical Informatics (SMI) Group. The added value of the latter was that it provided support to the ontology merge method Prompt.

Approximately at the same time, the AIFB Institute of the University of Karlsruhe developed the **FCA-Merge toolset** [107] to support the FCA-Merge method.

Finally, in 2002, **GLUE** [23] was developed at the University of Washington. GLUE is a system that semi-automatically finds mappings between concepts from two different ontologies.

The current ontology merging approaches have the following deficiencies: (1) mappings to perform the merging are usually established by hand; (2) all the tools need the participation of the user to obtain a definitive result in the merging process; and (3) no tool allows the merging of axioms and rules. The natural evolution of merging tools should lead to increased use of knowledge and to decreased participation of the people in the process. This could improve the possibilities of the merging at run-time.

In the context of the workshop on Evaluation of Ontology Tools EON2004, an experiment was performed on the quality of the mappings provided by different methods and tools. This will be continued in other efforts.

To learn more about ontology alignment and merging we recommend readers to access the Ontology Matching Web page.[17]

1.7.4 Ontology Evolution and Versioning

Ontologies are often developed by several groups of people and may evolve over time. Therefore, they cannot be understood as static entities, but rather are dynamic ones. As a consequence, ontology versioning becomes necessary and essential. This support must enable users to compare versions of ontologies and analyze differences between them [93]. Ontology engineers working in parallel on the same ontology need to maintain and compare different versions, to examine the changes that others have performed, and to accept or reject the changes. Ontology-based application developers should easily see the changes between ontology versions, determine which definitions were added or deleted, and accept or reject the changes. Let us note that, for ontologies, we must compare the semantics of the ontologies and not their serializations, since two ontologies that are exactly the same conceptually may have very different text representations when implemented in some ontology languages.

The **change management KAON plugin** allows the effects of changes through *evolution strategies* to be stablished [104]. A particular evolution strategy allows us to establish, for example, what happens with its sub-

[17] http://www.ontologymatching.org/

classes when a concept C is deleted: if they can also be deleted, or they can become subclasses of the superclasses of C.

The **PromptDiff algorithm** compares ontologies producing an initial set of mappings between two versions of the same ontology [93]. For instance, if a term t_1 of the version v_1 has the same type as the term t_2 of the version v_2 (both of them are concepts, both of them are properties, etc.) and t_1 has a similar name to t_2, it is assumed that the semantics of t_1 and t_2 are similar. Therefore, t_1 and t_2 are mapped as similar terms. This initial set of mappings is propagated using a fixed-point algorithm that combines the results of the previous step. Thus, for example, if all the siblings of the concept C_1 of v_1 are mapped with siblings of the concept C_2 of v_2, C_1 and C_2 are candidates to be mapped through a change operation (e.g., the addition of a new subclass). This algorithm is implemented by the **PromtDiff API** (see Sect. 1.7.3).

1.7.5 Ontology Evaluation

Work on ontology content evaluation started in 1994 [35]. In the last few years, interest in this issue has grown and extended to the evaluation of technology used to build ontologies. A survey of evaluation methods and tools can be found in [39]. These evaluation efforts can be examined under the following four perspectives.

From a ***content perspective***, many libraries exist where ontologies are published and publicly available (DAML,[18] KAON,[19] Ontobroker,[20] Ontolingua,[21] Protégé,[22] SemWebCentral,[23] SHOE,[24] WebODE,[25] WebOnto,[26] etc.). No documentation is available about how ontologies available in libraries or well-known and large ontologies (e.g., Cyc [72], or Sensus [109]) were evaluated. However they have been used to build many successful applications.

From a ***methodology perspective***, the main efforts to evaluate ontology content were made by Gómez-Pérez [40, 36] in the framework of

[18] http://www.daml.org/ontologies/
[19] http://kaon.semanticweb.org/
[20] http://ontobroker.semanticweb.org/
[21] http://www-ksl-svc.stanford.edu:5915/
[22] http://protege.stanford.edu/
[23] http://semwebcentral.org/index.jsp/
[24] www.cs.umd.edu/projects/plus/SHOE/onts/index.html
[25] http://webode.dia.fi.upm.es/
[26] http://webonto.open.ac.uk/

METHONTOLOGY, and by Guarino and colleagues [114] with the OntoClean method.

Gómez-Pérez has identified and classified different kinds of errors in taxonomies. Such identification can be used as a checklist for taxonomy evaluation. Such a list presents a set of possible errors that can be made by ontology engineers when modeling taxonomic knowledge in an ontology under a frame-based approach. Errors are classified as: inconsistency, incompleteness and redundancy errors. The ontology engineer should not postpone the evaluation until the taxonomy is finished; the control mechanisms should be performed during construction of the taxonomy.

OntoClean is a method elaborated by the Ontology Group of the CNR in Padova (Italy). Its goal is to remove wrong *Subclass-Of* relations in taxonomies according to some philosophical notions such as *rigidity*, *identity* and *unity*. According to this method, the ontology engineer firs, assigns some meta-properties to each concept of the taxonomy (e.g., if each instance of the concept is a whole, then it applies a set of rules that establish the possible incompatibilities of values in the taxonomy). Such rules allow pruning of the wrong *subclass of* links if the values assigned to a concept are incompatible with the values assigned to its children.

From an ***implementation perspective***, we can find important connections and implications between the components we use to build ontologies (concepts, relations, properties and axioms); the KR paradigms (frames, description logics, first-order logic, and so on); and the languages we use to implement them. This is important because different KR paradigms offer different reasoning mechanisms that we can use in content evaluation (e.g., description logic classifiers, or frame-based reasoning).

From a ***technological perspective***, ontology tool developers have gained experience evaluating tools working on the OntoWeb European thematic network SIG3 (Special Interest Group on Enterprise Standard Ontology Environments). Different ontology tool developers have also conducted comparison studies of different types of ontology tools, which can be found in the OntoWeb deliverable D1.3 [37]. According to these studies, evaluation functionalities of well-known ontology development tools (Protégé, WebODE, OntoEdit, etc.) allow the checking of taxonomies. However, such evaluation functionalities are still not enough for a deep ontology evaluation.

Recently, some researchers have published a synthesis of their experience in ontology evaluation [19, 38, 48, 90]. According to their conclusions, although good ideas have been provided in this area, there are still important deficiencies. Other interesting works are those in [51] and the above-mentioned EON2004 experiment.

1.7.6 Ontology Implementation

The implementation activity (proposed by all the methods and methodologies, and supported by all the development tools) consists of building computable models in an ontology language. As stated in the introduction, two groups of languages can be identified: classical and markup. We recommend [39] for detailed descriptions of each of them, where the same ontology is implemented in each language. Now, we briefly describe the most relevant ones.

KIF [34] is a language based on first-order logic created as an interchange format for diverse KR systems. Ontolingua [43, 27], which builds on KIF, combines the KR paradigms of frames and first order predicate calculus (KIF). It is the most expressive of all the languages that have been used for representing ontologies, allowing the representation of concepts, taxonomies of concepts, *n*-ary relations, functions, axioms, instances and procedures. Its high expressiveness led to difficulties in building reasoning mechanisms for it.

Loom [75] was not initially meant for implementing ontologies, but for general KBs. Loom is based on description logics (DL) and production rules, and provides automatic classifications of concepts. The following ontology components can be represented with this language: concepts, concept taxonomies, *n*-ary relations, functions, axioms and production rules. This language has nom been superseded by Powerloom.

OCML [87] was created as a kind of operational Ontolingua. In fact, most of the definitions that can be expressed in OCML are similar to the corresponding definitions in Ontolingua, and some additional components can be defined: deductive and production rules, and operational definitions for functions. OCML was built for developing executable ontologies and models in PSM.

FLogic [66] (*Frame Logic*) combines frames and first-order logic, allowing the representation of concepts, concept taxonomies, binary relations, functions, instances, axioms and deductive rules. FLogic is the only one of the previous languages that do not have Lisp-like syntax. Any of its inference engines, OntoBroker [21] or FLORA [73], can be used for constraint checking and deducing new information.

The OKBC (*Open Knowledge Base Connectivity*) protocol [15] (which is not properly a language) allows access to KBs stored in different knowledge representation systems (KRSs). Of the systems presented above, Ontolingua and LOOM are OKBC compliant.

SHOE [74] was built first as an extension of HTML and later as a language using the XML syntax. It uses different tags from those of the HTML specification, thus it allows the insertion of ontologies in HTML

documents. SHOE combines frames and rules. SHOE just allows the representation of concepts, their taxonomies, *n*-ary relations, instances and deduction rules, which are used by its inference engine to obtain new knowledge.

XOL [63] was developed as a XMLization of a small subset of primitives from the OKBC protocol, called OKBC-Lite. It is a very restricted language where only concepts, taxonomies and binary relations can be specified. No inference mechanisms are attached to it, as it was mainly designed for the exchange of ontologies in the biomedical domain.

RDF [70] was developed by the W3C (the World Wide Web Consortium) as a semantic network- based language to describe Web resources. Finally, the RDF Schema [13] language was also built by the W3C as an extension to RDF with frame-based primitives. The combination of both RDF and RDF Schema is normally known as RDF(S). RDF(S) is much less expressive than the previous languages, just allowing the representation of concepts, taxonomies of concepts and binary relations. Some inference engines have been created for this language, mainly for constraint checking.

These languages have established the foundations of the Semantic Web. In this context, three more languages have been developed as extensions to RDF(S): OIL, DAML+OIL and OWL.

OIL [54] added frame-based KR primitives to RDF(S), and its formal semantics was based on description logics. DAML+OIL [55] allows the representation concepts, taxonomies, binary relations, functions and instances. These two languages are currently no longer used.

Finally, in 2001, the W3C formed a working group called the Web-Ontology (WebOnt) Working Group.[27] The aim of this group was to devise a new ontology markup language for the Semantic Web, called OWL (Ontology Web Language). This language was proposed as a W3C recommendation in February 2004. Figure 1.5 shows how these languages have evolved, and it also shows the relationships of these languages with other existing KR languages and systems.

In the previous languages, only some of them are well equipped with primitives that allow exploitation of the concept of networked ontologies. These are Ontolingua, OCML, Flogic, RDF, RDF Schema and OWL (OIL and DAML+OIL also supported this notion, but they are no longer active). Based on our experience and on the case studies available from the literature, we have identified some associations between the ontology languages and the different kinds of ontology-based applications where they are applied.

[27] http://www.w3.org/2001/sw/WebOnt/

In e-commerce applications, ontologies are usually used for representing products and services that are offered on e-commerce platforms and are given to users in catalogues they can browse through [71]. Representational needs are not too complex: basically, we need concepts and attributes, and *n*-ary relations between concepts. However, reasoning needs are usually higher: if the number of products or services offered on the platform is high, automatic classifications are very useful for organizing these products or services automatically (hence, languages based on description logics are extremely helpful), and an efficient query answering is also important in this environment (this is provided by most of the studied languages).

When using PSMs and domain ontologies together two languages are strongly recommended, as they provide explicit support for this integration as well as reusable libraries: namely OCML and FLogic. In fact, both of them are operational modeling languages and solve the issue of PSM prototyping easily. A generic model of parametric design problem solving is provided in OCML [87], and KARL [28] (a customization of FLogic) has been used for PSM modeling, too.

In the context of the Semantic Web, and for exchanging ontologies between applications, languages based on XML are easily read and managed since standard libraries for the treatment of XML are available free. However, it is not difficult to adapt traditional languages to XML syntax, which could make use of the same kinds of libraries. The main advantage of RDF(S) and OWL is the strong support they receive from other communities besides the ontology community, and this means that more tools are available for editing, handling and documenting the ontologies.

The creation of upper-level ontologies requires high expressiveness and mostly there are not great needs for reasoning support. Upper-level ontologies have been generally specified in DL languages such as LOOM or CLASSIC. The Cyc KB is specified in CycL [72], which is a language based on frames and first order logic.

Some efforts are now being made now to migrate these ontologies to OWL.[28] In general, languages based on DL have been widely used in applications that needed intelligent integration of heterogeneous information sources. For instance, CLASSIC has been used in OBSERVER [81], LOOM in Ariadne [6], and OIL has been used in an urban planning process [105], among others. In addition, most of them have been used for information retrieval. For example, LOOM has been used in OntoSeek [49]. The main reason for this broad use is their inference support.

[28] http://www.w3.org/TR/2004/REC-owl-ref-20040210/

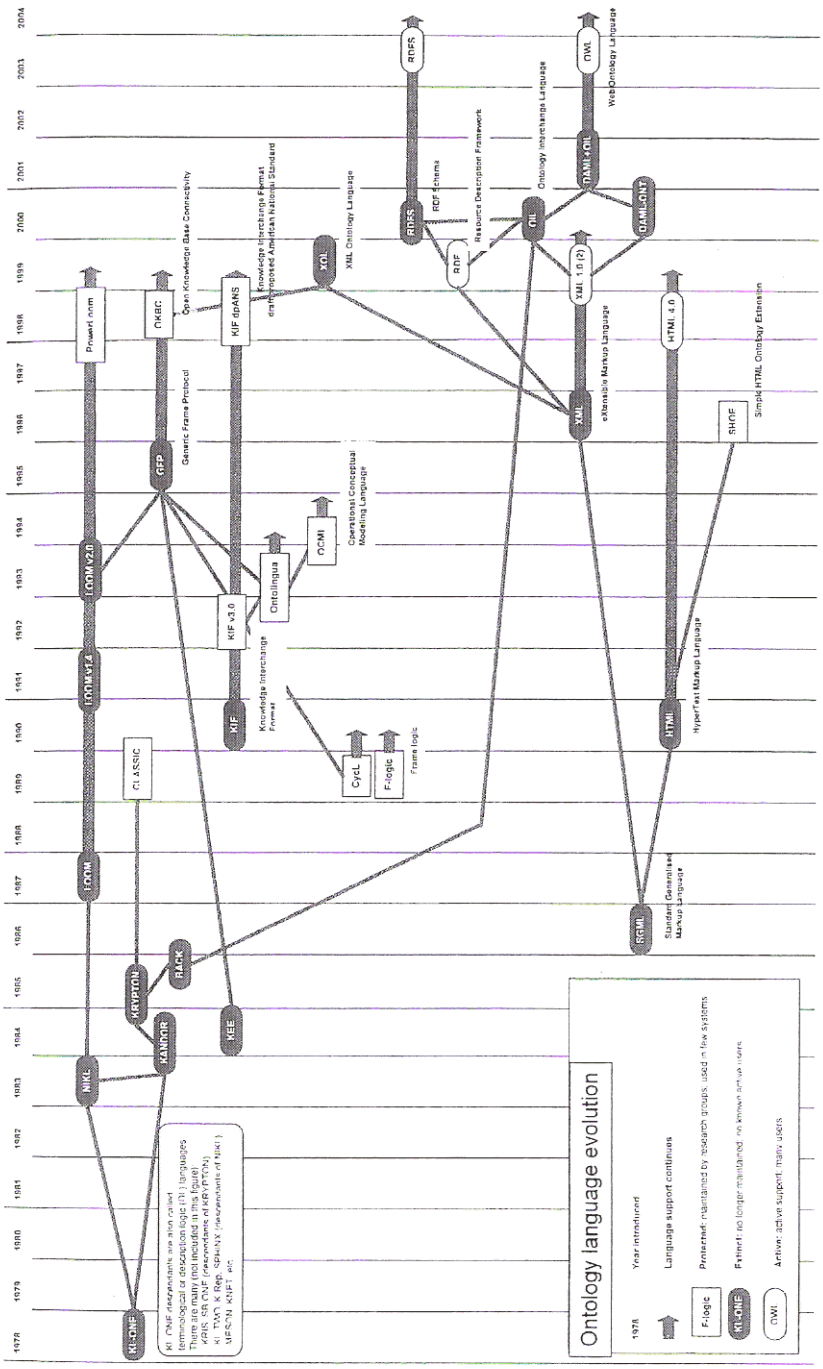

Fig. 1.5. Ontology language evolution

1.8 Conclusions

At the beginning of the 1990s ontology development was similar to an art: ontology developers did not have clear guidelines on how to build ontologies but only some design criteria to be followed. Work on principles, methods and methodologies, together with supporting technology, turned ontology development into engineering. This migration process was mainly due to the definition of the ontology development process and the ontology life cycle, which described the steps to be performed in order to build ontologies and the interdependencies among all those steps.

Though ontologies were clearly oriented to be reused, it has not been until recently, with the emergence of the Semantic Web, that appropriate support at all levels (methodologically and technologically, including implementation languages) has been provided. Many aspects of ontological engineering still need to be adapted to this situation. In this chapter we have reviewed existing ontology principles, methods and methodologies, tools, and languages, focusing especially on those that support the notion of networked ontologies, and on the new life cycle that appears as a consequence of this new framework. The following is a summary of the chaper.

Ontology engineers have available methodologies that guide them through the ontology development process. METHONTOLOGY is the methodology that provides the most detailed descriptions of the processes to be performed; On-To-Knowledge is the one that covers most activities, although with very short descriptions of processes; and the Grüninger and Fox methodology is the most formal one. All of them consider the reuse of existing ontologies during the development process, but only METHONTOLOGY has recently adapted its proposal for a life cycle to the environment of networked ontologies. In any case, the development activities are the most detailed in all of them, mainly the specification, conceptualization and implementation. There is still a lack of proposals for ontology management activities (scheduling, control and quality assurance), and for some pre-development (e.g., environment study) and post-development activities (e.g., (re)use).

Concerning support activities, some interesting contributions have been made in ontology learning, ontology merging and alignment, ontology evolution, and ontology evaluation, as described in Sect. 1.7. Nevertheless, important work has to be done in all of these activities. For example, the time when activities like ontology learning or ontology merging can be applied to heavyweight ontologies is still far away.

One of the problems that the ontology engineer can find when (s)he has to build an ontology is that (s)he has to use different methods that are not integrated. For example, ontology learning methods are not integrated in methodologies that cover the whole development process (e.g., in METHONTOLOGY or On-To-Knowledge). Some experience exists in the integration of methods in methodologies. For example, the OntoClean method has been integrated in METHONTOLOGY (see [30]).

A similar problem appears in the use of ontology tools, given that there is a lack of integrated environments for ontology development. Tools are usually created as isolated modules that solve one type of problem, but neither are fully integrated, nor do they interoperate with other tools that implement other activities of the ontology life cycle.

Finally, work on ontology languages has been constantly evolving since the first languages that were made available for ontology implementation, most of them based on existing KR languages. The existence of heterogeneous networked ontologies has been mainly considered in the recent language developments created in the context of the Semantic Web (RDF, RDF Schema and OWL), with the addition of namespaces that allow referring to ontology components that have been defined elsewhere and with the use of import primitives to include an existing model in an ontology.

1.9 Acknowledgements

This work has been partially supported by the IST project Knowledgeweb (FP6-507482) and by the Spanish project Semantic Services (TIN 2004-02660).

References

1. Arpírez JC, Corcho O, Fernández-López M, Gómez-Pérez A (2003) WebODE in a nutshell. AI Magazine 24(3): 37–38. Fall 2003
2. Arpírez JC, Gómez-Pérez A, Lozano A, Pinto HS (1998) (ONTO)^2 Agent: An ontology-based WWW broker to select ontologies. In: Gómez-Pérez A, Benjamins RV (eds) ECAI'98 Workshop on Applications of Ontologies and Problem-Solving Methods. Brighton, United Kingdom, pp. 16–24
3. Aussenac-Gilles N, Biébow B, Szulman S (2000a) Revisiting Ontology Design: A Methodology Based on Corpus Analysis. In: Dieng R, Corby O (eds) 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'00). Juan-Les-Pins, France. (Lecture Notes in

Artificial Intelligence LNAI 1937) Springer-Verlag, Berlin, Germany, pp. 172–188

4. Aussenac-Gilles N, Biébow B, Szulman S (2000b) Corpus analysis for conceptual modelling. In: Aussenac-Gilles N, Biébow B, Szulman S (eds) EKAW'00 Workshop on Ontologies and Texts. Juan-Les-Pins, France. CEUR Workshop Proceedings 51:1.1–1.8. Amsterdam, The Netherlands. http://CEUR-WS.org/Vol-51/
5. Aussenac-Gilles N, Seguela P (2000) Les relations sémantiques: du linguistique au formel. In : Condamines A (ed) Cahiers de grammaire, N° spécial sur la linguistique de corpus (Presse de l'UTM, Vol 25), Toulouse, pp. 175–198
6. Barish G, Knoblock GA, Chen Y-S, Minton S, Philpot A, Shahabi C (2000) The theaterloc virtual application. In: Engelmore R, Hirsh H (eds) 12th Annual Conference on Innovative Applications of Artificial Intelligence (IAAI'00). Austin, Texas, pp. 980–987
7. Bechhofer S, Horrocks I, Goble C, Stevens R (2001) OilEd: a reasonable ontology editor for the Semantic Web. In: Baader F, Brewka G, Eiter T (eds) Joint German/Austrian Conference on Artificial Intelligence (KI'01). Vienna, Austria. (Lecture Notes in Artificial Intelligence LNAI 2174) Springer-Verlag, Berlin, Germany, pp. 396–408
8. Beneventano D, Bergamaschi S, Castano S, Corni A, Guidetti R, Malvezzi G, Melchiori M, Vincini M (2000) Information integration: the MOMIS project demonstration. In: El Abbadi A, Brodie ML, Chakravarthy S, Dayal U, Kamel N, Schlageter G, Whang KY (eds) 26th International Conference on Very Large Data Bases, Cairo, Egypt. Morgan Kaufmann Publishers, San Francisco, California, pp. 611–614
9. Bernaras A, Laresgoiti I, Corera J (1996) Building and reusing ontologies for electrical network applications. In: Wahlster W (ed) European Conference on Artificial Intelligence (ECAI'96). Budapest, Hungary. John Wiley & Sons, Chichester, United Kingdom, pp. 298–302
10. Berners-Lee T (1999) Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor. HarperCollins Publishers, New York
11. Blázquez M, Fernández-López M, García-Pinar JM, Gómez-Pérez A (1998) Building Ontologies at the Knowledge Level using the Ontology Design Environment. In: Gaines BR, Musen MA (eds) 11th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'98). Banff, Canada, SHARE4:1–15
12. Bray T, Paoli J, Sperberg-McQueen CM, Maler E (2000) Extensible Markup Language (XML) 1.0. W3C Recommendation. http://www.w3.org/TR/REC-xml
13. Brickley D, Guha RV (2004) RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation. http://www.w3.org/TR/PR-rdf-schema

14. Castano S, De AntonellisV, De Capitani diVemercati S (2001) Global viewing of heterogeneous data sources. IEEE Transactions on Knowledge and Data Engineering 13(2):277–297
15. Chaudhri VK, Farquhar A, Fikes R, Karp PD, Rice JP (1998) Open Knowledge Base Connectivity 2.0.3. Technical Report.
http://www.ai.sri.com/~okbc/okbc-2-0-3.pdf
16. Chisholm R (1989) On Metaphysics. University of Minnesota Press, Minneapolis
17. Corcho O, Fernández-López M, Gómez-Pérez A, Vicente O (2002) WebODE: an Integrated Workbench for Ontology Representation, Reasoning and Exchange. In: Gómez-Pérez A, Benjamins VR (eds) 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02). Sigüenza, Spain. (Lecture Notes in Artificial Intelligence LNAI 2473) Springer-Verlag, Berlin, Germany, pp. 138–153
18. Corcho O, Gómez-Pérez A (2002) Ontology Translation Approaches for Interoperability: A Case Study with Protégé-2000 and WebODE. In: Motta E, Shadbolt N, Stutt A, Gibbins N (eds) 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02). (Lecture Notes in Artificial Intelligence LNAI 3257) Springer-Verlag, Berlin, Germany, pp. 30–46
19. Daelemans W, Reinberger ML (2004) Shallow Text Understanding for Ontology Content Evaluation. IEEE Intelligent Systems 19(4):76–78
20. Dean M, Schreiber G (2004) OWL Web Ontology Language Reference. W3C Recommendation.
http://www.w3.org/TR/owl-ref/
21. Decker S, Erdmann M, Fensel D, Studer R (1999) Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In: Meersman R, Tari Z, Stevens S (eds) Semantic Issues in Multimedia Systems (DS-8), Rotorua, New Zealand. Kluwer Academic Publishers, Boston, Massachusetts. pp. 351–369
22. Declerck T, Uszkoreit H (2003) State of the art on multilinguality for ontologies, annotation services and user interfaces. Esperonto deliverable D1.5.
http://www.esperonto.net
23. Doan A, Madhavan J, Domingos P, Halevy A (2002) Learning to Map between Ontologies on the Semantic Web. In: Lassner D (ed) Proceedings of the 11th International World Wide Web Conference (WWW 2002), Honolulu, Hawaii.
http://www2002.org/refereedtrack.html
24. Domingue J (1998) Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web. In: Gaines BR, Musen MA (eds) 11th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'98). Banff, Canada, KM4:1–20
25. Ehring M, Staab S (2004) QOM – Quick Ontology Mapping. In: McIlraith SA, Plexousakis D (eds) 3rd International Semantic Web Conference

(ISWC'04), Hiroshima, Japan. (Lecture Notes in Computer Science LNCS 3298) Springer-Verlag, Berlin, Germany, pp. 683–697
26. Euzenat J (2004) An API for Ontology Alignment. In: McIlraith SA, Plexousakis D (eds) 3rd International Semantic Web Conference (ISWC'04), Hiroshima, Japan. (Lecture Notes in Computer Science LNCS 3298) Springer-Verlag, Berlin, Germany, pp. 698–712
27. Farquhar A, Fikes R, Rice J (1997) The Ontolingua Server: A Tool for Collaborative Ontology Construction. International Journal of Human Computer Studies 46(6):707–727
28. Fensel D (1995) The Knowledge Acquisition and Representation Language KARL. Kluwer Academic Publishers, Boston, Massachusetts
29. Fernández-López M, Gómez-Pérez A (2002a) Overview and analysis of methodologies for building ontologies. The Knowledge Engineering Review 17(2):129–156
30. Fernández-López M, Gómez-Pérez A (2002b) The integration of OntoClean in WebODE. In: Angele J, Sure Y (eds) EKAW'02 Workshop on Evaluation of Ontology-based Tools (EON2002), Sigüenza, Spain. CEUR Workshop Proceedings 62:38–52. Amsterdam, The Netherlands. http://CEUR-WS.org/Vol-62/
31. Fernández-López M, Gómez-Pérez A, Juristo N (1997) METHONTOLOGY: From Ontological Art Towards Ontological Engineering. Spring Symposium on Ontological Engineering of AAAI. Stanford University, California, pp. 33–40
32. Fernández-López M, Gómez-Pérez A, Pazos A, Pazos J (1999) Building a Chemical Ontology Using Methontology and the Ontology Design Environment. IEEE Intelligent Systems 4(1):37–46
33. Gangemi A, Pisanelli DM, Steve G (1999) An Overview of the ONIONS Project: Applying Ontologies to the Integration of Medical Terminologies. Data & Knowledge Engineering 31(2):183–220
34. Genesereth MR, Fikes RE (1992) Knowledge Interchange Format. Version 3.0. Reference Manual. Technical Report Logic-92-1. Computer Science Department, Stanford University, California. http://meta2.stanford.edu/kif/Hypertext/kif-manual.html
35. Gómez-Pérez A (1994) Some Ideas and Examples to Evaluate Ontologies. Knowledge Systems Laboratory, Stanford University, California. http://www-ksl.stanford.edu/KSL_Abstracts/KSL-94-65.html
36. Gómez-Pérez A (2001) Evaluation of Ontologies. International Journal of Intelligent Systems 16(3):391–409
37. Gómez-Pérez A (2002) A Survey on Ontology Tools, OntoWeb deliverable D1.3. http://ontoweb.aifb.uni-karlsruhe.de/About/Deliverables/D13_v1-0.zip
38. Gómez Pérez A (2004) Evaluating ontology evaluation. IEEE Intelligent Systems 19(4):74–76
39. Gómez-Pérez A, Fernández-López M, Corcho O (2003) Ontological Engineering. Springer–Verlag, London, United Kingdom

40. Gómez-Pérez A, Fernández-López M, de Vicente A (1996) Towards a method to conceptualize domain ontologies. In: van der Vet P (ed) ECAI'96 Workshop on Ontological Engineering. Budapest, Hungary, pp. 41–52
41. Gómez-Pérez A, Manzano D (2003) A survey of ontology learning methods and techniques. OntoWeb deliverable D.1.5. http://www.ontoweb.org
42. Gómez-Pérez A, Rojas MD (1999) Ontological Reengineering and Reuse. In: Fensel D, Studer R (eds) 11th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW'99). Dagstuhl Castle, Germany. (Lecture Notes in Artificial Intelligence LNAI 1621) Springer-Verlag, Berlin, Germany, pp. 139–156
43. Gruber TR (1992) Ontolingua: A Mechanism to Support Portable Ontologies. Technical report KSL-91-66, Knowledge Systems Laboratory, Stanford University, Stanford, California. ftp://ftp.ksl.stanford.edu/pub/KSL_Reports/KSL-91-66.ps
44. Gruber TR (1993a) A translation approach to portable ontology specification. Knowledge Acquisition 5(2):199–220
45. Gruber TR (1993b) Toward principles for the design of ontologies used for knowledge sharing. In: Guarino N, Poli R (eds) International Workshop on Formal Ontology in Conceptual Analysis and Knowledge Representation. Padova, Italy. (Formal Ontology in Conceptual Analysis and Knowledge Representation) Kluwer Academic Publishers, Deventer, The Netherlands. http://citeseer.nj.nec.com/gruber93toward.html
46. Grüninger M, Fox MS (1995) Methodology for the design and evaluation of ontologies. In: Skuce D (ed) IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, pp. 6.1–6.10
47. Guarino N (1998) Formal Ontology in Information Systems. In: Guarino N (ed) 1st International Conference on Formal Ontology in Information Systems (FOIS'98). Trento, Italy. IOS Press, Amsterdam, The Nederlands, pp. 3–15
48. Guarino N (2004) Toward Formal Evaluation of Ontology Quality. IEEE Intelligence Systems 19(4):78–79
49. Guarino N, Masolo C, Vetere G (1999) OntoSeek: Content-Based Access to the Web. IEEE Intelligent Systems 14(3):70–80
50. Guarino N, Welty C (2002) Evaluating Ontological Decisions with OntoClean. Communications of the ACM 45(2):61–65
51. Guo Y, Pan Z, Heflin J (2004) An Evaluation of Knowledge Base Systems for Large OWL Datasets. In: McIlraith SA, Plexousakis D (eds) 3rd International Semantic Web Conference (ISWC'04), Hiroshima, Japan. (Lecture Notes in Computer Science LNCS 3298) Springer-Verlag, Berlin, Germany, pp. 274–288
52. Hermenegildo M, Bueno F, Cabeza D, Carro M, García M, López P, Puebla G (2000) The Ciao Logic Programming Environment. In: Lloyd JW, Dahl V, Furbach U, Kerber M, Lau K, Palamidessi C, Pereira LM, Sagiv Y, Stuckey PJ (eds) International Conference on Computational Logic (CL'00).

London, United Kingdom. (Lecture Notes in Computer Science LNCS 1861), Springer-Verlag, Berlin, Germany
53. de Hoog R (1998) Methodologies for Building Knowledge Based Systems: Achievements and Prospects. In: Liebowitz J (ed) Handbook of Expert Systems. CRC Press, Boca Raton, Florida, Chapter 1.
54. Horrocks I, Fensel D, Harmelen F, Decker S, Erdmann M, Klein M (2000) OIL in a Nutshell. In: Dieng R, Corby O (eds.) 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'00). Juan-Les-Pins, France. (Lecture Notes in Artificial Intelligence LNAI 1937) Springer-Verlag, Berlin, Germany, pp. 1–16
55. Horrocks I, van Harmelen F (eds) (2001) Reference Description of the DAML+OIL (March 2001) Ontology Markup Language. Technical Report. http://www.daml.org/2001/03/reference.html
56. Hustadt U, Motik B, Sattler U (2004) Reducing SHIQ Description Logic to Disjunctive Datalog Programs. In: Williams MA (ed) 9th International Conference on the Principles of Knowledge Representation and Reasoning (KRR'04). Whistler, Canada, pp. 152–162
57. IEEE (1996) IEEE Standard for Developing Software Life Cycle Processes. IEEE Computer Society, New York. IEEE Std 1074-1995
58. Joachims T (1998) A probabilistic analysis of the Rocchio Algorithm with TFIDF for text categorization. In: Fisher DH (ed) 14th International Conference on Machine Learning (ICML'97). Nashville, Tennessee. Morgan Kaufmann Publishers, San Francisco, California, pp. 143–151
59. Johansson I (1989) Ontological Investigations. Routledge, New York
60. Kalfoglou Y, Robertson D (1999a) Use of Formal Ontologies to Support Error Checking in Specifications. In: Fensel D, Studer R (eds) 11th European Workshop on Knowledge Acquisition, Modelling and Management (EKAW'99), Dagsthul, Germany. (Lecture Notes in Artificial Intelligence LNAI 1621) Springer-Verlag, Berlin, Germany, pp. 207–224
61. Kalfoglou Y, Robertson D (1999b) Managing Ontological Constraints. In: Benjamins VR, Chandrasekaran A, Gómez-Pérez A, Guarino N, Uschold M (eds) IJCAI99 Workshop on Ontologies and Problem-Solving Methods (KRR-5), Stockholm, Sweden. CEUR Workshop Proceedings 18:5.1–5.13. Amsterdam, The Netherlands. http://CEUR-WS.org/Vol-18/
62. Kalyanpur A, Parsia B, Hendler J (2005) A Tool for Working with Web Ontologies. International Journal of Semantic Web and Information Systems 1(1):36–49
63. Karp PD, Chaudhri V, Thomere J (1999) XOL: An XML-Based Ontology Exchange Language. Version 0.3. Technical Report. http://www.ai.sri.com/~pkarp/xol/xol.html
64. Khan L, Luo F (2002) Ontology Construction for Information Selection. In: Ramamoorthy CV (ed.) 14th IEEE International Conference on Tools with Artificial Intelligence. Washington DC, pp. 122-127

65. Kietz JU, Maedche A, Volz R (2000) A Method for Semi-Automatic Ontology Acquisition from a Corporate Intranet. In: Aussenac-Gilles N, Biébow B, Szulman S (eds) EKAW'00 Workshop on Ontologies and Texts. Juan-Les-Pins, France. CEUR Workshop Proceedings 51:4.1–4.14. Amsterdam, The Netherlands.
http://CEUR-WS.org/Vol-51/
66. Kifer M, Lausen G, Wu J (1995) Logical Foundations of Object-Oriented and Frame-Based Languages. Journal of the ACM 42(4): 741–843
67. Klein M, Fensel D (2001) Ontology versioning on the Semantic Web. In: Cruz IF, Decker S, Euzenat J, McGuinness DL (eds) First International Semantic Web Workshop (SWWS'01). Stanford, California
68. Klein M, Fensel D, Kiryakov A, Ognyanov D (2002) Ontology versioning and change detection on the Web. In: Gómez-Pérez A, Benjamins VR (eds) 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02). Sigüenza, Spain. (Lecture Notes in Artificial Intelligence LNAI 2473) Springer-Verlag, Berlin, Germany, pp. 197–212
69. Knublauch H, Fergerson R, Noy NF, Musen MA (2004) The Protege OWL Plugin: An Open Development Environment for Semantic Web Applications. In: McIlraith SA, Plexousakis D (eds) 3rd International Semantic Web Conference (ISWC'04), Hiroshima, Japan. (Lecture Notes in Computer Science LNCS 3298) Springer-Verlag, Berlin, Germany, pp. 229–243
70. Lassila O, Swick R (1999) Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation.
http://www.w3.org/TR/REC-rdf-syntax/
71. Léger A, Arbant G, Barrett P, Gitton S, Gómez-Pérez A, Holm R, Lehtola A, Mougenot I, Nistal A, Varvarigou T, Vinesse J (2000) MKBEEM: Ontology domain modeling support for multilingual services in e-Commerce. In: Benjamins VR, Gómez-Pérez A, Guarino N, Uschold M (eds) ECAI'00 Workshop on Applications of Ontologies and PSMs. Berlin, Germany, pp. 19.1–19.4
72. Lenat DB, Guha RV (1990) Building Large Knowledge-based Systems: Representation and Inference in the Cyc Project. Addison-Wesley, Boston, Massachusetts
73. Ludäscher B, Yang G, Kifer M (2000) FLORA: The Secret of Object-Oriented Logic Programming.
http://www.cs.sunysb.edu/~sbprolog/flora/docs/manual.ps
74. Luke S, Heflin JD (2000) SHOE 1.01. Proposed Specification. Technical Report. Parallel Understanding Systems Group, Department of Computer Science, University of Maryland.
http://www.cs.umd.edu/projects/plus/SHOE/spec1.01.htm
75. MacGregor R (1991) Inside the LOOM classifier. SIGART Bulletin 2(3):70–76
76. Madhavan J, Bernstein PA, Rahm E (2001) Generis schema matching with Cupid. In: Apers PMG, Atzeni P, Ceri S, Paraboschi S, Ramamohanarao K, Snodgrass RT (eds) 27th International Conference on Very Large Data

Bases. Rome, Italy Morgan Kaufmann Publishers, San Francisco, California, pp. 49–58
77. Maedche A, Motik B, Stojanovic L, Studer R, Volz R (2003) Ontologies for Enterprise Knowledge Management. IEEE Intelligent Systems 18(2):26–33
78. Maedche A, Staab S (2000) Semi-automatic Engineering of Ontologies from Text. In Chang SK, Obozinski WR (eds) 12th International Conference on Software Engineering and Knowledge Engineering (SEKE'2000). Chicago, Illinois
79. McGuinness D, Fikes R, Rice J, Wilder S (2000) The Chimaera Ontology Environment. In: Rosenbloom P, Kautz HA, Porter B, Dechter R, Sutton R, Mittal V (eds) 17th National Conference on Artificial Intelligence (AAAI'00). Austin, Texas, pp. 1123–1124
80. Melnik S, García-Molina H, Rahm E (2002) Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In: Georgakopoulos D (ed) 18th International Conference on Data Engineering ICDE'2002. San José, California, pp. 117–128
81. Mena E, Kashyap V, Sheth AP, Illarramendi A (1996) OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. In: Litwin W (ed) First IFCIS International Conference on Cooperative Information Systems (CoopIS'96). Brussels, Belgium, pp. 14–25
82. Mikheev A, Finch A (1997) A Workbench for Finding Structure in Texts. In: Grishman R (ed) 5th Applied Natural Language Processing Conference (ANLP'97). Washington, DC
83. Miller GA (1995) WordNet: a lexical database for English. Communications of the ACM 38(11):39–41
84. Mitra P, Wiederhold G, Kersten (2000) A graph-oriented model for articulation of ontology interdependencies. In: Lockemann PC (ed) 7th International Conference on Extending Database Technology, EDBT 2000. Lecture Notes in Computer Science 1777, Springer-Verlag, Berlin, Germany, pp. 86–100
85. Morin E (1998) Prométhée un outil d'aide a l'acquisition de relations semantiques entre temes. In: Zweigenbaum P (ed) 5ème National Conference on Traitement Automatique des Langues Naturelles (TALN'98). Paris, France, pp. 172–181
86. Morin E (1999) Acquisition de patrons lexico-syntaxiques caractéristiques dúne relation sémantique. TAL (Traitement Automatique des Langues) 40(1):143–166
87. Motta E (1999) Reusable Components for Knowledge Modelling: Principles and Case Studies in Parametric Design. IOS Press, Amsterdam, The Netherlands
88. Neches R, Fikes RE, Finin T, Gruber TR, Senator T, Swartout WR (1991) Enabling technology for knowledge sharing. AI Magazine 12(3):36–56
89. Newell A (1982) The Knowledge Level. Artificial Intelligence 18(1):87–127
90. Noy NF (2004) Evaluation by Ontology Consumers. IEEE Intelligent Systems 19(4):80–81

Noy NF, Fergerson RW, Musen MA (2000) The knowledge model of Protege-2000: Combining interoperability and flexibility. In: Dieng R, Corby O (eds) 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'00). Juan-Les-Pins, France. (Lecture Notes in Artificial Intelligence LNAI 1937) Springer-Verlag, Berlin, Germany, pp. 17–32

Noy NF, Klein M (2002) Ontology Evolution: Not the Same as Schema Evolution. Technical Report SMI-2002-0926, Stanford, California. http://smi-web.stanford.edu/pubs/SMI_Abstracts/SMI-2002-0926.html

Noy NF, Kunnatur S, Klein M, Musen MA (2004) Tracking Changes During Ontology Evolution. In: McIlraith SA, Plexousakis D (eds) 3rd International Semantic Web Conference (ISWC'04), Hiroshima, Japan. (Lecture Notes in Computer Science LNCS 3298) Springer-Verlag, Berlin, Germany, pp. 259–273

Noy NF, Musen MA (2000) PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In: Rosenbloom P, Kautz HA, Porter B, Dechter R, Sutton R, Mittal V (eds) 17th National Conference on Artificial Intelligence (AAAI'00). Austin, Texas, pp. 450–455

Noy NF, Musen MA (2001) Anchor-PROMPT: Using Non-Local Context for Semantic Matching. In: Gómez-Pérez A, Grüninger M, Stuckenschmidt H, Uschold M (eds) IJCAI'01 Workshop on Ontologies and Information Sharing. Seattle, Washington, pp. 63–70

Noy NF, Musen MA (2004a) Specifying Ontology Views by Traversal. In: McIlraith SA, Plexousakis D (eds) 3rd International Semantic Web Conference (ISWC'04), Hiroshima, Japan. (Lecture Notes in Computer Science LNCS 3298) Springer-Verlag, Berlin, Germany, pp. 713–725

Noy NF, Musen MA (2004b) Ontology versioning in an ontology-management framework. IEEE Intelligent Systems 19(4):6–13

Pan R, Ding Z, Yu Y, Peng Y (2005) A Bayesian Network Approach to Ontology Mapping. In: 4th International Semantic Web Conference (ISWC'05). Galway, Ireland. (Lecture Notes in Computer Science LNCS 3729) Springer-Verlag, Berlin, Germany, pp. 563–577

Raggett D, Le Hors A, Jacobs I (1999) HTML 4.01 Specification. W3C Recommendation. http://www.w3.org/TR/html401/

Schreiber ATh, Wielinga BJ, Jansweijer W (1995) The KACTUS View on the 'O' World. In: Skuce D (ed) IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, pp. 15.1–15.10

Shvaiko P, Giunchiglia F, Yatskevich M (2004) S-Match: an Algorithm and an Implementation of Semantic Matching. In: Fensel D, Studer R (eds) 1st European Semantic Web Symposium (ESWS'04). Heraklion, Greece. (Lecture Notes in Computer Science LNCS 3053) Springer-Verlag, Berlin, Germany, pp. 61–75

Staab S, Schnurr HP, Studer R, Sure Y (2001) Knowledge Processes and Ontologies. IEEE Intelligent Systems 16(1):26–34

103. Steve G, Gangemi A, Pisanelli DM (1998) Integrating Medical Terminologies with ONIONS Methodology. In: Kangassalo H, Charrel JP (eds) Information Modeling and Knowledge Bases VIII. IOS Press, Amsterdam, The Netherlands.
http://ontology.ip.rm.cnr.it/Papers/onions97.pdf
104. Stojanovic L (2004) Methods and Tools for Ontology Evolution. PhD Thesis, FZI, Karlsrhue, Germany
105. Stuckenschmidt H (2000) Using OIL for Intelligent Information Integration. In: Benjamins VR, Gómez-Pérez A, Guarino N (eds) ECAI'00 Workshop on Applications of Ontologies and Problem Solving Methods. Berlin, Germany, pp. 9.1–9.10
106. Studer R, Benjamins VR, Fensel D (1998) Knowledge Engineering: Principles and Methods. IEEE Transactions on Knowledge and Data Engineering 25(1–2):161–197
107. Stumme G, Maedche A (2001) FCA-MERGE: Bottom-Up Merging of Ontologies. In: Bernhard Nebel (ed) Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001). Seattle, Washington. Morgan Kaufmann Publishers, San Francisco, California, pp. 225–234
108. Sure Y, Erdmann M, Angele J, Staab S, Studer R, Wenke D (2002) OntoEdit: Collaborative Ontology Engineering for the Semantic Web. In: Horrocks I, Hendler JA (eds) First International Semantic Web Conference (ISWC'02). Sardinia, Italy. (Lecture Notes in Computer Science LNCS 2342) Springer-Verlag, Berlin, Germany, pp. 221–235
109. Swartout B, Ramesh P, Knight K, Russ T (1997) Toward Distributed Use of Large-Scale Ontologies. In: Farquhar A, Gruninger M, Gómez-Pérez A, Uschold M, van der Vet P (eds) AAAI'97 Spring Symposium on Ontological Engineering. Stanford University, California, pp. 138–148
110. Thomasson AL (2004) Methods of Categorization. Invited Talk. In: Varzi AC, Vieu L (eds) Proceedings of 3rd Formal Ontoloy in Information Systems, Turin, Italy, pp. 3–16
111. Uschold M (1996) Building Ontologies: Towards A Unified Methodology. In: Watson I (ed) 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems. Cambridge, United Kingdom.
http://citeseer.nj.nec.com/uschold96building.html
112. Uschold M, Grüninger M (1996) Ontologies: Principles, Methods and Applications. Knowledge Engineering Review 11(2):93–155
113. Uschold M, King M (1995) Towards a Methodology for Building Ontologies. In: Skuce D (eds) IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal, Canada, pp. 6.1–6.10
114. Welty C, Guarino N (2001) Supporting Ontological Analysis of Taxonomic Relationships. Data and Knowledge Engineering 39(1):51–74
115. Wu SH, Hsu WL (2002) SOAT: A Semi-Automatic Domain Ontology Acquisition Tool from Chinese Corpus. In: Lenders W (ed) 19th International Conference on Computational Linguistics (COLING'02), Taipei, Taiwan

Calero · Ruiz · Piattini (Eds.)

Ontologies for Software Engineering and Software Technology

Communication is one of the main activities in software projects, many such projects fail or encounter serious problems because the stakeholders involved have different understandings of the problem domain and/or they use different terminologies. Ontologies can help to mitigate these communication problems.

Calero and her coeditors mainly cover two applications of ontologies in software engineering and software techonology: sharing knowledge of the problem domain and using a common terminology among all stakeholders, and filtering the knowledge when defining models and metamodels.

The editors structured the contributions into three parts: first, a detailed introduction into the use of ontologies in software engineering and software technology in general; second, the use of ontologies to conceptualize different process-related domains such as software maintenance, software measurement, or SWEBOK, initiated by IEEE; third, the use of ontologies as artifacts in several software processes, like, for example, in OMG's MOF or MDA.

By presenting the advanced use of ontologies in software research and software projects, this book is of benefit to software engineering researchers in both academia and industry.

ISBN 3-540-34517-5
90000
9 783540 345176

springer.com